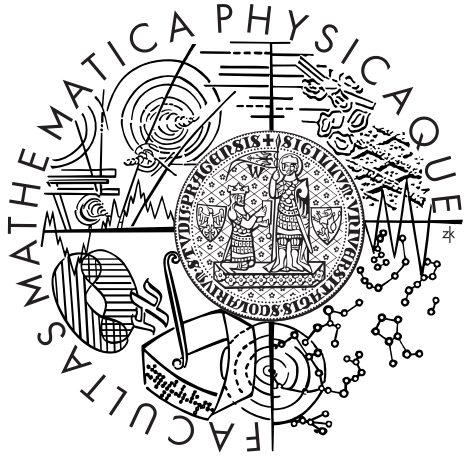
Charles University in Prague
Faculty of Mathematics and Physics

**MASTER THESIS**




Jan Mašek

# Detection and Correction of Inconsistencies in the Multilingual Treebank HamleDT

Ústav formální a aplikované lingvistiky

Supervisor of the master thesis: Zdeněk Žabokrtský

Study programme: Computer Science

Specialization: Mathematical Linguistics

Prague 2014

I would like to thank my advisor, doc. Ing. Zdeněk Žabokrtský, Ph.D., for introducing me to HamleDT, his guidance and valuable advice.

I would like to thank my evaluators and proofreaders – Vojta Diatka, Ondra Dušek, Hana Gabrielová, Michal Lázníčka, Radek Ocelák, Aničkа Plasová, Jitka Šípková, and Zuzka Vaňková.

Most of all, I would like to thank my wife, Martina Mašek Panešová, for her support, patience, and her unconditional love.

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In ..... date .....  signature of the author

Název práce: Detection and Correction of Inconsistencies in the Multilingual Treebank HamleDT

Autor: Jan Mašek

Ústav: Ústav formální a aplikované lingvistiky

Vedoucí diplomové práce: doc. Ing. Zdeněk Žabokrtský, Ph.D., Ústav formální a aplikované lingvistiky

Abstrakt: Prostudovali jsme závislostní korpusy, jež jsou součástí projektu HamleDT, a částečně jsme sjednotili soubor značek užitých pro anotaci syntaktické roviny. Následně jsme použili metodu založenou na *variačních n-gramech* pro automatickou detekci chyb na morfologické a syntaktické rovině. Potom jsme využili výstup morfologického značkovače, respektive závislostního syntaktického analyzátoru pro opravení chyb detekovaných v předchozím kroku. Spolehlivost detekce i opravy chyb na obou anotačních rovinách jsme vyhodnotili na základě náhodně vybraných vzorků nalezených předpokládaných chyb z několika korpusů.

Klíčová slova: závislostní korpusy, detekce chyb, oprava chyb, variační $n$-gramy

Title: Detection and Correction of Inconsistencies in the Multilingual Treebank HamleDT

Author: Jan Mašek

Department: Institute of Formal and Applied Linguistics

Supervisor: doc. Ing. Zdeněk Žabokrtský, Ph.D., Institute of Formal and Applied Linguistics

Abstract: We studied the treebanks included in HamleDT and partially unified their label sets. Afterwards, we used a method based on *variation n-grams* to automatically detect errors in morphological and dependency annotation. Then we used the output of a part-of-speech tagger / dependency parser trained on each treebank to correct the detected errors. The performance of both the detection and the correction of errors on both annotation levels was manually evaluated on a randomly selected samples of suspected errors from several treebanks.

Keywords: dependency treebanks, error detection, error correction, variation $n$-grams

# Contents

# 1. Introduction

Morphologically and syntactically annotated corpora constitute an important resource for both traditional and computational linguistics, and provide valuable data for a variety of tasks in natural language processing, including parsing, machine translation, information retrieval, and others. However, there is no universally accepted annotation scheme and many treebanks use their own. This makes it difficult to conduct experiments on multiple languages, which is in some cases (machine translation, cross-lingual parsing, ...) necessary, and in many others useful. HamleDT[1] is a collection of about thirty dependency treebanks created in part to mitigate this problem – the treebanks are *harmonized* to follow the same annotation scheme. A researcher thus needs to understand just one scheme to be able to experiment on many different languages.

The quality of corpora is vital, because errors in annotation can lead to worse results – but even the highest quality corpora contain errors. One of the ways of decreasing the errors is manual post-annotation checking. However, manual work is expensive and (given a low proportion of errors) ineffective, providing diminishing returns. Also, no human post-annotator is infallible; he is likely to correct only some occurrences of an error while overlooking others – reducing the number of errors, but possibly creating inconsistencies.

A partial solution is to somehow detect the errors automatically with relatively high precision. It is then possible to let the human annotator inspect only the much smaller set containing the detected (suspected) errors, improving the efficiency of the process a great deal. However, while manual correction of detected errors is a good idea in general, it is unfeasible for HamleDT because of its scope – it would be difficult (and, again, expensive) to find reliable annotators for many of the languages. A more viable way would be to correct the errors automatically as well.

Based on the review of the relevant literature, the most promising way for the automatic detection of errors seems to be the *variation n-grams* method of Dickinson and Meurers (2003a). It offers high precision; and when we detect errors reliably enough, the subsequent correction should improve the corpus even if the correction method has a lower precision (which we expect, as it is in principle more difficult to correct an error than to find it). In the hypothetical extreme case of 100 % detection precision, every change in the correction step replaces an error – either with a correct annotation (which reduces the number of errors by one), or with another error (which keeps the number of errors the same).

Experiments with the detection of errors in part-of-speech, phrase-structure or dependency annotation using the variation $n$-grams method were conducted on corpora of English, German, Swedish and Czech. (e.g. Dickinson and Meurers, 2003a; Dickinson, 2005; Boyd et al., 2008) We apply the variation $n$-grams method to detect the errors in both the part-of-speech and dependency annotation of the treebanks of 30 languages in HamleDT including many non-Germanic and non-Indo-European languages.

[1] HArmonized Multi-LanguagE Dependency Treebank, `http://ufal.mff.cuni.cz/hamledt`

We then use a tagger to automatically correct the detected errors in part-of-speech annotation (following Dickinson (2005)), and extend this method to automatically correct the detected errors in dependency annotation using a dependency parser. A manual evaluation of a sample of the detected errors and of the proposed corrections reveals that the method works successfully in the sense that it can be expected to always reduce the number of errors (or at least not to increase it). The results conform to our expectation that the method performs better on the morphological than on the syntactic level.

We are of the opinion that it is worth to apply the part-of-speech annotation corrections; the case of dependency annotation is less clear cut, as the benefit of the reduced number of errors might be in fact offset by a decrease in consistency. A preferable option might be to provide the corrections as an optional patch to the data in the same way as the harmonized annotation for the non-free treebanks in HamleDT is provided.

Examination of the detected errors and proposed correction also provides some insight into possible issues in harmonization in general, and in harmonization of different grammatical constructions in individual treebanks in particular.

## 1.1 Structure of the Thesis

The remainder of the thesis is structured as follows. Chapter 2 gives an overview of other authors' works on various methods for automatic detection or correction of errors in morphological or syntactical annotation, including the *variation n-grams* method which forms the core of this work. Chapter 3 describes the data used in our experiments and our preprocessing of the data. Chapter 4 starts by the explanation of the *variation n-grams* method devised by Dickinson and Meurers (2003a). The rest of the chapter is divided in two parts dealing with the morphological level (Section 4.2) and the syntactic level (Section 4.3). Each of them first presents the method as it is implemented for the respective level in scripts from project DECCA[2] that we use, and then our implementation of a method for the correction of errors using an existing tagger and parser, respectively. Chapter 5 contains the summarization of the results we obtained in our experiments, their evaluation, and discussion. Chapter 6 concludes the thesis.

[2]Detection of Errors and Correction in Corpus Annotation, `http://decca.osu.edu/`

# 2. Related work

This chapter provides a brief overview of other authors' works related to automatic detection and/or correction of errors in morphological (part-of-speech) and/or syntactic annotation. It is divided into three sections: Section 2.1 describes several different methods used for the morphological level. Section 2.2 deals with various methods applicable to syntactic annotation, with focus on dependency trees. And finally, Section 2.3 summarizes the body of work experimenting with *variation n-grams* method for both levels, upon which we build in this thesis.

## 2.1 Part-of-speech Annotation

Eskin (2000) works with a (presumably language independent) method based on *anomaly detection* to automatically detect part-of-speech annotation errors in the Penn Treebank corpus. He uses a *mixture model* (Barnett and Lewis, 1994), which assumes the data are generated by two probability distributions (one for the "normal" elements, and one for the anomalies). The distributions are modeled by sparse Markov transducers or naive Bayes trained on all of the data. When "the difference between the log likelihood of the distribution if the element is removed from the majority distribution and included in the anomalous distribution" is "sufficiently large", the element is considered an anomaly. When applied on the Penn Treebank (with 1.25 million tagged elements), the method detected 4000 anomalies with a precision[1] of 44 %. The method might be useful for a subsequent manual correction,[2] but because of its relatively low precision, any automatic method for correction would have to solve the problem of many false positives. The anomaly detection method also cannot detect errors arising from inter-annotator inconsistencies, and not every error is manifested as an anomaly. (Eskin, 2000)

Loftsson (2009) also focuses on automatic detection of part-of-speech annotation errors. He experiments with three different methods on the Icelandic Frequency Dictionary: The first is the *variation n-grams* method described by Dickinson and Meurers (2003a; see also Sections 2.3 and 4.1). The second method is to "run five different taggers on the corpus and examine those cases where all the taggers agree on a tag, but, at the same time, disagree with the gold standard annotation". Both of these methods are language independent, in contrast to the third one,[3] which is based on the development of various patterns based on feature agreement in the shallow parses of the sentences in the corpus. The three methods are complementary – a large proportion of the (true) errors

[1] In the context of automatic error detection, we define precision as follows: precision = true errors/(true errors + false errors), where *true error* means "the method was correct in marking the token as an error" and *false error* the opposite, i.e. "the method was mistaken in marking the token as an error".

[2] Especially given the ranking of the detected errors – for the first 1000 errors, the precision rises to 69 %.

[3] However, Loftsson argues that "the method can be adapted to other morphologically complex languages".

detected by each one are not detected by the other two. Loftsson does not compute the precision for the *variation n-grams* method; for the 5 taggers and the shallow parsing methods, the precision is 16.6 % and 30.1 %, respectively, which makes them unsuitable for a subsequent automatic correction.

## 2.2 Syntactic Annotation

### 2.2.1 Phrase Structure

Kato and Matsubara (2010) propose "a method for correcting annotation errors in a [phrase structure] treebank" using a *synchronous tree substitution grammar* automatically induced from the treebank to transform erroneous parse trees into correct ones. Using the method on the Penn Treebank, they achieve the precision of 71.9 % on the 100 highest ranked rules (corresponding to 331 "tokens"); 70 of those achieved the precision of 100 %.

### 2.2.2 Dependency

B. Agrawal et al. (2013) present a method for detection of both attachment and labeling errors in a dependency treebank using a parser error patterns to build a knowledge base. They aim for a high recall with lower regard for precision, because they assume a subsequent manual correction and argue that "[a] human validator can reject unintuitive errors without much effort". The idea is to incorporate the automatic error detection and their manual correction into the treebank development. In experiments on the Hindi Dependency Treebank (Husain et al., 2010), the method detects errors in dependency annotation with a precision of 64.58 % and a recall of 88.57 %. However, it must be noted that inter-chunk dependency trees were used. The authors claim that the method is language independent, however, for each treebank, the knowledge base must be constructed separately, making it impractical for a large collection of different treebanks.

Haverinen et al. (2011) examine the errors in the dependency annotation in the Turku Dependency Treebank (Haverinen et al., 2010), which uses a modified version of the Stanford Dependencies (Marneffe and Manning, 2008). This Finnish treebank is exceptional by having used *full double annotation*, which means that "each sentence is first independently annotated by two different annotators, and all differences are then jointly resolved, creating the *merged* annotation". The authors first investigate which dependency labels and constructions are "the most difficult for the human annotators and the baseline parser" by comparing the differences between the *merged* annotation and the *individual* annotations. More relevant, they train a binary SVM classifier to distinguish correctly and incorrectly annotated tokens and use the results to rank the sentences. "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." (The "random baseline" means choosing the sentences at random; one would expect a random sample of 25 % of the sentences to contain 25 % of all annotation errors.) The

intention is to "significantly reduce the amount of double annotation [...] needed in a compromise setting where full double annotation is not possible".

Volokh and Neumann (2011) use a language independent method to automatically detect and correct errors in dependency annotation in the English dependency corpus (Marcus et al., 1993). They train "two different state of the art parsers: the graph-based MSTParser [(McDonald et al., 2005)] and the transition-based MaltParser [(Nivre et al., 2006)]" and then parse the training data. The cases where both parsers disagree with the gold standard are considered potential errors. These are then substituted with the MSTParser predictions and a third parser (MDParser) is used to parse the modified corpus; where "the modified annotation is identical with the one predicted by the third parser", the modification is kept.

Out of 6743 error candidates detected, 3535 were considered true errors and corrected. The authors assume the "method has a very high precision", although the exact number was not reported (as it could "probably be computed only by manual investigation of all corrected dependencies"); in the case all the corrections were justified, the detection precision would be 52.42 %. The recall was estimated at 45.9 %.

Novák and Razímová (2009) focus on the automatic detection of errors in the tectogrammatical (deep syntactic) layer of the Prague Dependency Treebank (Hajič et al., 2006). Their method is "based on Apriori algorithm [(R. Agrawal and Srikant, 1994)] for mining association rules from data sets". They use it to extract "the highly confident rules [...] and find the annotations where these rules are violated", thus getting the list of anomalies – possible errors. In an experiment, the authors found that out of 100 most suspicious cases, 20 were true errors (so the precision can be assumed to be bounded from above at approximately 20 %).

## 2.3 Variation $n$-grams Method

Dickinson and Meurers (2003a) show two language specific methods – closed class analysis and finite-state tagging guide patterns – but more importantly, propose the (non-language-specific) *variation n-grams* method for "detecting errors in 'gold-standard' part-of-speech annotation". The method is "based on $n$-grams occurring in the corpus with multiple taggings" – such $n$-grams are considered possible errors. As the method forms the core of this thesis, it is described in detail later in Section 4.1. On the Wall Street Journal corpus, out of 2495 variation nuclei types found in $n$-grams of length between 6 and 224, 2436 were true errors.

Dickinson and Meurers (2003b) extend the *variation n-grams* method to detecting errors in (phrase-structure) syntactic annotation. Dickinson (2005) discusses the method in depth, extending it even further to detecting errors in discontinuous constituents in syntactic annotation; he experiments with a tagger-based method for correction of part-of-speech annotation errors and concludes with identifying the automatically correctable errors. Boyd et al. (2007) explore the possibility of increasing the recall of error detection in phrasal treebanks by using part-of-speech $n$-grams instead of word $n$-grams. They also

develop some new heuristics for increasing precision. Boyd et al. (2008) extend the *variation n-grams* method to detect errors in dependency annotation, and Dickinson (2009) experiments with memory-based learning for automatic correction of dependency errors. In addition to words and tags, he uses ambiguity classes as features and constraints, reaching a correction precision of 76.7 %.

# 3. Data

The data used in this thesis are all part of HamleDT – *HArmonized Multi-LanguagE Dependency Treebank* – which "is a compilation of existing dependency treebanks (or dependency conversions of other treebanks), transformed so that they all conform to the same annotation style". (Zeman et al., 2012; Zeman et al., 2014) HamleDT currently contains treebanks of thirty languages. Twenty-one of them are Indo-European: five Slavic (Bulgarian [bu], Czech [cs], Russian [ru], Slovak [sk] and Slovenian [sl]), five Germanic (Danish [da], Dutch [nl], English [en], German [de] and Swedish [sv]), Latin [la] and five Romanic (Catalan [ca], Italian [it], Portuguese [pt], Romanian [ro] and Spanish [es]), Greek ancient [el] and modern [grc], and the Indo-Iranian Persian [fa], Bengali [bn], and Hindi [hi]). The others include Arabic [ar], Basque [eu], three Uralic languages (Estonian [et], Finnish [fi], and Hungarian [hu]), Japanese [ja], Tamil [ta], Telugu [te], and Turkish [tr].

Throughout this work, we follow the HamleDT convention of denoting the treebanks by the ISO 693-1 codes of their languages as listed above; and unless explicitly stated otherwise (or undoubtedly clear from the context), by using either the code or the name of a language, we refer just to the treebank of the language, not to the language itself. For the list of all the source treebanks with references and information about their sizes see Attachment A.

In the rest of this chapter, we first in Section 3.1 concisely present the harmonization process and the Prague Dependencies annotation style, and then remark upon the differences from the Prague Dependency Treebank and some problematic labels in Section 3.2. Section 3.3 closes this chapter by the description of our data preprocessing.

## 3.1 Harmonization and Prague Dependencies

The treebanks included in HamleDT have been *harmonized* to conform with the Prague Dependencies style (Rosa et al., 2014), which is derived from the annotation style of the Prague Dependency Treebank (Hajič, 1998; Hajič et al., 2006; Bejček et al., 2013). A few notable differences between the two styles are described in Section 3.2. The harmonization process is implemented in Treex,[1] which is a modular NLP framework written in Perl. In Treex, any task is "decomposed into a sequence of subsequent steps [...] called *blocks*". (Popel and Žabokrtský, 2010)

During a typical harmonization, the original treebank is first converted into UTF-8-encoded Treex XML format. The rest of the conversion is then done via a Treex block `HamleDT::<upper-case ISO 639-1 language code>::Harmonize`, which converts the part-of-speech tags to the Interset, and the dependency labels and structure to the Prague Dependencies style. A good overview of the harmonization process can be found in Zeman et al. (2014).

[1] `http://ufal.mff.cuni.cz/treex`

### 3.1.1 Morphology

Morphological (part-of-speech) tags are converted from the original tagset into Interset (Zeman, 2008) via a tagset-specific *driver*. Interset is intended as a kind of universal morphological tagset – to "provide a unified representation for as many feature values in existing tagsets as possible". (Zeman et al., 2014) Each tag is represented as a set of feature–value pairs.

The basic feature is `pos` (part of speech) with the following possible values:

- `noun` – noun
- `adj` – adjective
- `num` – numeral
- `verb` – verb
- `adv` – adverb
- `adp` – adposition
- `conj` – conjunction
- `part` – particle
- `int` – interjection
- `punc` – punctuation
- `sym` – symbol

Note that there is no specific part of speech for pronouns and/or articles and (pre)determiners; they are instead distributed among the nouns, adjectives and other classes on the basis of their syntactic properties, and recognizable by having an appropriate value of the `prontype` (pronoun type) feature.

In total, there are about 50 different features in the Interset. Apart from `tagset` (which identifies the source tagset driver) and `other` (which can store any information), each feature has a predefined set of possible values. The features range from the most typical ones like `case`, `number`, `tense`, `aspect` or `mood` that are in one form or the other present in most of the languages, to the more exotic ones like `politeness` or `echo` ("is this a reduplicative or echo word?").[2]

### 3.1.2 Dependency Structure

There are many linguistic structures for which there is no single clearly correct representation. Drawing on Zeman et al. (2014), we present a few examples of the decisions taken in the Prague Dependency Treebank and by extension in the Prague Dependencies style of HamleDT.

#### Coordinations

The coordinations are arguably one of the most difficult structures to capture in a dependency tree. In HamleDT, they are represented using the so-called Prague family, where one of the conjunctions is the head, and all conjuncts are siblings depending on it. Even inside this family, there is much variation;

[2]For a comprehensive list of features and their possible values, see `https://wiki.ufal.ms.mff.cuni.cz/user:zeman:interset:features`

using the classification by Popel et al. (2013), we can say that in the Prague Dependencies, the rightmost conjunction is chosen for the head, the other conjunctions (including punctuation) and the shared modifiers are attached to the coordination head, the relation of the conjuncts to the parent of the coordination is expressed by their labels, and both conjuncts and shared modifiers are annotated. For a complex example, see Figure 3.1.

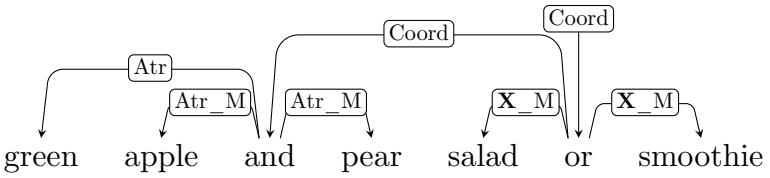



**Figure 3.1:** Coordination style in Prague Dependencies.
(The label suffix `_M` marks a conjunct.)

### Other Structures

In adpositional phrases, the adposition is the head; it bears a special label AuxP, and its child bears the label that expresses the relation of the whole adpositional phrase to its parent. A simple example can be seen in Figure 3.2, which also shows the representation of determiners – they are governed by the noun.

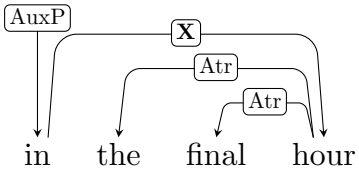



**Figure 3.2:** Adpositional phrase and determiner in Prague Dependencies.

Subordinate clauses are represented in a similar way. If there is a subordinating conjunction, it is the head with a special label AuxC, and its child (the root of the subordinate clause) bears the label expressing the relation of the clause to its parent; in case no subordinating conjunction is present, the root of the subordinate clause is attached to the word it modifies directly – see Figure 3.3 and Figure 3.4.

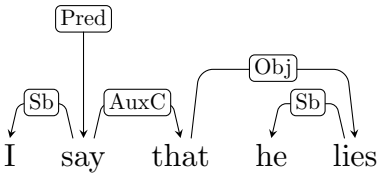



**Figure 3.3:** Subordinated clause with a conjunction.

There is yet no unified approach to the harmonization of verb groups, although some basic principles should be followed: the auxiliary verbs in periphrastic constructions depend on the main verb with a label AuxV; the modal

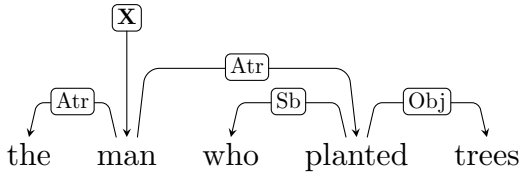



**Figure 3.4:** Subordinated clause without a conjunction.

and phase verbs are usually the head of the whole verb group with a label representing the relation of the verb group to its parent, and they govern the content verb in the infinitive with a label Obj; the subject and negative particles usually depend on the head, while other arguments are attached to the content verb.

### 3.1.3 Dependency Labels

There were 42 distinct labels (excluding the AuxS of the technical root) that appeared in at least one of the treebanks in HamleDT 1.5. This number has been reduced to 31 in HamleDT 2.0 and further to 25 in the current (as of yet unpublished) version. About 15 of them are present in most or all treebanks. Several more probably should, from the linguistic standpoint, be present in (almost) all treebanks, but are not; it may be impossible to distinguish them based on the annotation in the original treebanks, or there might be an error in the harmonization process. Finally, the rest of the labels, which are present in only some of the treebanks, mark relations that are distinguishable in only some of the languages or treebanks, or that are somewhat more peripheral and easily lend themselves to a different interpretation. The borderline cases between some of the labels are sometimes hard to decide even for an experienced annotator of Czech; and there are no language-specific annotation guidelines for other languages.

It must be also mentioned that the status of some of the labels – namely AuxA (article or determiner) and Neg (negative particle) – is not yet fixed. They have been used in the harmonization of some of the languages, but not others – in HamleDT 2.0, they are present in 10 and 2 treebanks, respectively. In the current version, AuxA remains just in [ta] and once in [ro], and Neg has remained in [eu] and [ro] and was added in [hu]. In other treebanks, the relations are either subsumed under Atr and Adv, respectively, or they are not applicable at all for linguistic reasons, as some languages do not have articles or express negation by bounded morphemes. It has not yet been decided for either of the two labels whether they should be kept in the label set, or whether they should be removed completely; either decision will require a modification of the harmonization of some languages.

Most of the dependency labels express the relation of the node, or rather of the subtree rooted in the node, to its parent; the exceptions ar Pred, Coord, AuxP, AuxC and ExD. The following 21 labels are inherited from PDT:

- Pred – main predicate, a node not depending on another node
- Sb – subject
- Obj – object

- Adv – adverbial
- Atv – (determining) complement; a node with a dual dependency (a verb and one of its arguments), hung on the argument
- AtvV – as Atv, but hung on the verb because the nominal parent is missing in the tree
- Atr – attribute
- Pnom – nominal predicate
- AuxV – auxiliary verb
- Coord – coordination head
- AuxT – reflexive tantum
- AuxR – reflexive (neither Obj nor AuxT), passive reflexive
- AuxP – primary adposition or a part of secondary adposition
- AuxC – subordinating conjunction
- AuxO – redundant or emotional item, 'coreferential' pronoun
- AuxZ – focalizer
- AuxX – non-coordination-head comma
- AuxG – other punctuation and symbols
- AuxY – "adverbs, particles not classed elsewhere"
- AuxK – sentence-final punctuation and other symbols
- ExD – "externally-dependent"; a technical label used when the "true" parent is elided

And the following four labels are newly added in HamleDT:

- Apposition – second part of appositional construction
- AuxA – article or determiner
- Neg – negative particle
- NR – technical value for a node whose correct label could not have been determined during the harmonization

## 3.2 Differences between PDT and Prague Dependencies

While the Prague Dependencies in HamleDT are very similar to their ancestor, a few notable differences exist. On the structural level, the main difference is the treatment of appositions. In PDT, they are captured analogically to the coordination structures – the connecting element (usually a comma) is the head of the apposition structure and gets the label Apos, and the adordinated[3]

[3]Adordination is a paratactical construction similar to coordination, but in contrast to coordination, the adordination members have identical referents. (cf. Daneš et al., 1987)

members themselves have a label which expresses their relation to the parent of the whole structure. In contrast, in HamleDT, the apposition is regarded as a hypotactical construction: the second member is governed by the first, and has the label Apposition. A different label is presumably used to highlight this difference in structure.

There are also some differences in the label set. The four labels mentioned above (Apposition, AuxA, Neg, NR) have been added. However, NR is basically a label for an inadequacy in the harmonization, so it should not appear at all in the "final" version. The questionable status of AuxA and Neg has been described in Section 3.1.3.

Some other labels have been removed. As well as the already mentioned Apos, HamleDT label set does not include "combined" labels AtrAtr, AtrAdv, AdvAtr, AtrObj and ObjAtr, which are used in PDT for cases of "structural ambiguity [...] without semantic difference" between the two labels from which they are composed. (Hajič et al., 2004)

### 3.2.1 Problematic labels

The labels for the core sentence constituents, namely Pred, Sb, Obj, Adv and Atr, usually do not pose much of a problem in themselves as far as harmonization is concerned; they are attested in all of the treebanks. The only other such label is the coordination head Coord. However, the boundaries between them and some other labels are much less clear; most other labels also suffer from the fact that some of the more specific relations may not be recognized in some annotation styles. In this section, we offer several examples of such problems.

One might expect the adposition label AuxP to be missing in some of the treebanks, as not every language uses adpositions – their typical function, the case marking, may be expressed for example by bounded morphemes. However, the only treebank with no occurrence of AuxP is [eu]. This seems to be an error in the harmonization, as according to Aranzabe et al. (2003), the Basque language does have adpositions. In the annotation style of the original treebank, adpositions are governed by noun and marked with a label ncmod, the same as most other non-clausal adnominal modifiers, which is probably the reason for this oversight – most of them get assigned the label Atr.

Somewhat problematic are the labels Atv and AtvV. They have a dual dependency, which cannot be directly represented in a tree. Therefore, one of the dependencies is only implied, and the *verbal attribute* is hung on the parent that is lower in the tree, i.e. the nominal one, with the label Atv; only if the nominal parent is not explicitly represented in the tree, the verbal attribute is hung on the verb, but with a different label – AtvV – to distinguish the two cases. We believe that this distinction is not very useful, because in the more complex cases of Atv, is is impossible to (automatically) determine the verbal parent anyway, and the presence of two different labels for a single relation is confusing. Moreover, if a similar relation is represented in some of the original treebanks, it is likely that it will actually be represented as depending on the verb, as that is arguably its "main" dependency.

At least one of these labels is present in fifteen treebanks; both labels together appear in five, Atv in fourteen, and AtvV in six. We have not examined

the issue in depth, as it is out of the scope of this work, but we consider it very unlikely that all occurrences of the verbal attribute in a treebank would be cases with missing nominal parent as the labeling in [pt] suggests; we also suspect that in at least some of the treebanks where only Atv is present, it is actually (incorrectly) hung on the verbal parent.

The labels for punctuation – AuxX for comma, AuxK for sentence-end punctuation, and AuxG for other punctuation and symbols – are also missing in some of the languages. For [ro] and [ru], this is not an error in harmonization, but a feature of the original treebanks, as they do not contain punctuation at all. In [bn] and [te], some of the commas are assigned labels they are not allowed to have (e.g. some of the "core" labels).

Relatively complicated is the situation for some of the Aux_ labels in competition with each other and either adverbs (for AuxO, AuxY, AuxZ) or verb arguments (AuxO, AuxR, AuxT), and the results in the harmonized treebank heavily depend on the original label set and the author of the harmonization block, as there are no universal guidelines other than for Czech.

Another unclear situation arises around nominal predicate label Pnom. Some languages allow a nominal predicate without a copula and others do not; often it is unclear which verbal argument should be considered the subject and which the nominal part of the predicate; the set of copular verbs is construed with a different breadth in each language, so in the one extreme, every potential nominal part of a predicate may be labeled as an object, an in the other, any light verb may take Pnom as its argument.

## 3.3 Preprocessing

Before attempting to automatically detect and correct some of the inconsistencies and errors in HamleDT using the method outlined in Chapter 4, we had to prepare the data. This involved first "cleaning" the set of dependency labels by modifying the appropriate harmonization blocks, reharmonizing the treebanks to reflect the latest changes, and then file-format conversion and automatic generation of several files for each treebank, as described in Sections 3.3.1 and 3.3.2.

Some of the languages were using labels not present (and not really applicable) in any of the other treebanks, or labels that were removed from the HamleDT label set. The first category is represented by [ar] and [ta]. The Arabic treebank contained six such "endemic" labels. Three of them (PredC, PredE and PredP) represent various specific types of predicates, and we convert them to Pred; AuxE and AuxM represent "emphasizing expression" and "modifying expression", respectively, and we convert them to AuxZ; and finally, Ante stands for anteposition and is converted to Apposition.

The treebank of Tamil contains five such labels. Labels AAdjn and AComp represent adverbial adjunct and adverbial complement; these are both clear cases of adverbials distinguished only by the valency frame of the verb which governs them, so they are both converted to Adv. The remaining three labels are AdjAtr marking an adjectival participial or adjectivalized verb, Comp marking a non-adverbal complement, and CC for a separable part of a word. The former two are converted to Atr, and the latter one to AuxT (reflexivum tantum) if it

depends on a verb, and to Atr otherwise.

The second category consists of the combined labels described in Section 3.2 – AtrAtr, AtrAdv, AdvAtr, AtrObj, ObjAtr. According to the annotation manual, these should always hang on the possible parent that is lower in the tree, which will almost always be nominal in character. The other possible parent cannot be reliably determined. Therefore, the combined labels have been all converted to Atr; that means all of them hang correctly and no nodes have to be moved. The downside is that the information about the ambiguity is lost, and in cases where the verbal parent was preferred (i.e. AdvAtr and ObjAtr), the node was disambiguated into the less-preferred option. However, we believe that it is more than offset by the increased consistency across the treebanks (the labels were used just in [ar], [cs], [sk], and [sl], with little hope to appear anywhere else) and the improved label set.

### 3.3.1 For POS Detection and Correction

The script used in the detection of inconsistencies in the part-of-speech annotation, as well as the part-of-speech tagger used in the correction step, require the treebank to be in the TnT format (each line contains a single word form and its part-of-speech tag, separated by a tabulator, and the sentences are separated by a blank line). We have written the Treex block Print::TnT, which takes the files in the Treex format and prints them in the TnT format. For the part-of-speech tag, there are five options: 1. the `conll_pos` attribute, 2. the `conll_cpos` attribute, 3. the `conll_feat` attribute, 4. the Interset `pos`, or 5. the concatenation of Interset features without `tagset` and `other`. We ran the scenario "`Read::Treex Print::TnT`" on each of the treebanks with the last option.

To improve the correction results, we then assign *complex ambiguity tags* to the relevant words; the procedure is described in Section 4.2.3.

Apart from the training data in the TnT format, the tagger requires two more files – a lexicon, and a tagset. The lexicon file contains all word types from the corpus with all the part-of-speech tags with which they occur (with the exception regarding the cardinal numbers expressed by digits; the lexicon must contain just one such number). The tagset file contains all the tags that occur in the corpus. Technically, one could use the Interset to obtain the list of all possible tags including the ones not seen in the corpus; however, because we both train and run the tagger on the whole corpus, there are no unseen tags even if we just generate the list from the corpus. We have written a script (`make_DTT_lexicon.pl`) for the automatic creation of the lexicon file, and the tagset file is generated using a primitive bash script ("`cat` *`corpus`* `| cut -f2 | sort | uniq | tr '\n' '␣'`").

### 3.3.2 For Dependencies Detection and Correction

The script used in the detection of inconsistencies in the dependency annotation requires the corpus in a specific format and one more file described later in this paragraph. The process to obtain them is as follows. We first con-

vert the treebanks from the Treex format into the CoNLL format[4] using an existing Treex block `Write::CoNLLX` with options "`deprel_attribute='afun' pos_attribute='tag' cpos_attribute='iset/pos' feat_attribute='iset'`", and then use the script `CoNLL2Decca.py`[5] to convert them further into the Decca-XML[6] format. After that, we run the script `deccaxml-xsltproc.py` with `deccaxml-idwordposhead.xsl` stylesheet to finally transform the treebank into the required format – on each line, there is the id, word form, part-of-speech, and dependency head of the word, and sentences are separated by a single blank line. The other file that is needed must contain all potential *variation nuclei* (defined in Section 4.1) stored in a trie data structure. We first obtain a simple list of the possible variation nuclei by running `deccaxml-nuclei.py` with `deccaxml-nuclei.xsl`, and then create the file with the trie using `triefilter-idwordpos.py`.

In the correction step, we use a dependency parser, which requires the treebanks in MST format.[7] We use the script `conll2mst.py` (which is bundled with the parser) to convert the treebanks to this format from the CoNLL format (which we obtained in the preprocessing step for the part-of-speech annotation part).

[4]See Attachment B.1 for a detailed description.

[5]This script, and the scripts and stylesheets used to transform the treebanks to the required format and to create the nuclei and filtertries files, are provided by the project DECCA, `http://decca.osu.edu/`.

[6]`http://decca.osu.edu/schema/Decca-XML.xsd`

[7]See Attachment B.2 for a detailed description.

# 4. Method

In this chapter, we first present the basic idea of the variation $n$-grams method in Section 4.1. Section 4.2 contains the description of the algorithm and the heuristic we use for the detection of errors on the morphological level. It also includes the explanation of the complex ambiguity tags we use, and of the process of tagging and of the subsequent correction. Analogically, Section 4.3 first discusses the algorithm and the heuristic we use for the detection of errors on the syntactic level, and then the process of parsing and of the subsequent correction.

We use the *variation n-grams* method to automatically detect error candidates in both part-of-speech annotation and dependency annotation of the treebanks included in HamleDT. Subsequently, we train a decision-tree-based part-of-speech tagger, the TreeTagger (Schmid, 1994), and a non-projective dependency parser, the MSTParser (McDonald et al., 2005), on the entirety of each treebank, and then run them on the same data. We then use their output to decide for every error candidate whether it was in fact annotated correctly, or whether it is a true error, in which case we propose the parser/tagger output as a correct annotation. A sample of the proposed corrections is then manually evaluated for several languages.[1] The results of the detection and correction, and their manual evaluation can be found in Chapter 5.

## 4.1 Variation $n$-grams

Dickinson and Meurers (2003a) first proposed the *variation n-grams* method for detection of errors in part-of-speech annotation and later extended it to syntactic annotation of phrase-structure treebanks (2003b) and also dependency (2008) treebanks.

*Variation* refers to the situation where multiple *tokens* of the same *type* occur in the corpus annotated differently. A type is an abstraction, covering all the occurrences of an "annotation instance" in a corpus, while a token is a single occurrence – a word form with its part-of-speech tag in the part-of-speech annotation, and a pair of words together with a – possibly nonexistent – labeled dependency edge between them in the dependency annotation.

The variation in itself does not signify an error; although a single token usually has a single correct label, most types are ambiguous – they usually have a larger set of possible correct labels. For example, the word "meče" may in Czech be either a noun as in the sentence

(1) Rytíř se chopil *meče*.
knight seized sword.GEN.SG

(here in the genitive case of the singular – but the same form could in a different context also be the nominative or the accusative of the plural), or it may be

[1] The software used in the experiments, selected intermediate data and the results are available as a `tar.gz` archive and also on the enclosed DVD, whose contents are described in Attachment C.

a verb in a (present) transgressive form as in the sentence

(2) Kozel *meče* utekl.
billy-goat bleat.TGPRS ran-away
'Billy-goat ran away bleating.'

There are two possible causes for variation in a corpus annotation: either the type is ambiguous and the individual tokens realize its different variants; or the labeling of the type is inconsistent – the individual tokens with different labels realize the same variant and should have the same label, but some of them are annotated erroneously. The problem then becomes one of deciding whether each particular variation is a consequence of an ambiguity, or of an erroneous annotation.

As famously noted by Firth (1957), "[y]ou shall know a word by the company it keeps". We look at the context of the variation: a dissimilar context suggests the variation might be an ambiguity, whereas a similar context points to a possible annotation error. Because we intend to use this detection method to select error candidates for a subsequent automatic correction of a gold-standard corpora, we are concerned mostly with precision, with a lower regard to recall. Thus we follow Dickinson and Meurers (2003a) and look at the context composed of the word forms preceding and following the variation, and consider only identical contexts to be "similar". (One might instead look at e.g. part-of-speech tags of the surrounding words or at the dependency context – the dependency parent and children of the word in question.) The word that exhibits the variation is called *variation nucleus*, and an $n$-gram (of word forms) containing a variation nucleus – in other words, the variation nucleus together with its context – is called *variation $n$-gram*.

For example, the following 16-gram appears twice in the corpus.[2] in each case the word *processed* has a different tag – once `pos=adj|degree=pos`, and once `pos=verb|verbform=part|tense=past|aspect=perf`. The word *processed* is thus a variation nucleus of a variation 16-gram shown in (3); it is a single variation nucleus type which is represented by two variation nucleus tokens in the treebank.

(3) today , would apply to pesticides and other substances found on fresh and *processed* foods ,

## 4.2 POS

As mentioned above, in the case of part-of-speech annotation, we consider an annotation instance to be the form of the word together with its part-of-speech tag.

We work with a large set of typologically diverse languages, and the tagsets used by different treebanks vary greatly. One possible approach might be to work with the original tags of each treebank, and encode them into the Interset only after the error detection and correction. However, we decided to go a

[2] In the examples taken from HamleDT, we preserve the technical tokenization; the tokens include e.g. punctuation or parts of contracted words – for example, "don't" is split into two tokens, "do" and "n't". Every token is separated by a space.

different route and use the Interset directly. In our opinion, the main advantage of this is that the results for different treebanks are directly comparable; a beneficial side effect is that we do not have to learn the native tagsets of the evaluated treebanks, instead reusing the effort put into it by the authors of their Interset drivers.

Therefore, as the part-of-speech tags, we use a `|`-delimited concatenation of the Interset feature-value pairs[3] in the canonical order, as it is returned by `Treex::Core::Node::Interset::get_iset_conll_feat()`, with the exclusion of the `other` and `tagset` features. We treat the tags as atomic units without the internal structure they actually have. For more information on the tagset, see Section 3.1.1.

### 4.2.1 Algorithm

The algorithm used to obtain the list of all variation $n$-grams in a corpus is a modified variant of the Apriori algorithm (R. Agrawal and Srikant, 1994), originally proposed for association rules mining of transactional databases, adapted for this purpose by Dickinson and Meurers (2003a). It works in a bottom-up manner, in each iteration building the set of potential variation $n$-grams from the set of variation $(n-1)$-grams, based on the observation that each variation $n$-gram must contain a variation $(n-1)$-gram. In the algorithm below, $L_n$ is the set of all variation $n$-grams of length $n$ in the corpus, and $C_n$ is the set of variation $n$-grams candidates of length $n$ – in other words, the set of all $n$-grams containing an element of $L_{n-1}$.

1. Generate the set $C_1$ of variation unigrams candidates, i.e. of all unigrams in the corpus, and remember also their position in the corpus.
2. Prune $C_1$, leaving only the unigrams that exhibit variation, to obtain the set $L_1$ of all variation unigrams in the corpus.
3. Set $n = 1$.
4. Generate the set $C_{n+1}$ of $(n+1)$-grams candidates by extending the $n$-grams in $L_n$ to either side, remembering their positions in the corpus; stop if $C_{n+1} = \emptyset$.
5. Prune $C_{n+1}$, leaving only the $(n+1)$-grams that exhibit variation, to create the set $L_{n+1}$ of all variation $(n+1)$-grams in the corpus.
6. Increase $n$ by one and go to step 4.

We have used the RIDGES[4] modification of the DECCA project implementation in Python, which adds one postprocessing step after the algorithm itself: it goes once more through all the variation $n$-grams from unigrams up to the longest variation $n$-grams found, and removes from $L_n$ all $n$-grams that are contained in a $(n+1)$-gram in $L_{n+1}$, thus always keeping only the longest $n$-gram with the given variation nucleus.

[3]Some values may actually contain a disjunction of values delimited by a vertical bar – in that case, it is substituted by a semicolon.

[4]Register In Diachronic GErman Science, `http://korpling.german.hu-berlin.de/ridges/documentation-v4-en.html`

### 4.2.2 Heuristics

After obtaining all *variation n-grams* from a corpus, we can make use of some heuristics to divide them into (suspected) errors on one side and ambiguities on the other. Dickinson (2005) evaluates two heuristics for this purpose. The first one is "variation nuclei in long $n$-grams are errors", where for the purposes of the evaluation "long" means $n \geq 6$; the longer the identical context, the more likely it is that a variation signifies an error. The second one is "distrust the fringe", i.e. variation nucleus on the fringe (the beginning or the end) of the $n$-gram is probably not en error; this heuristic is based on the fact that morphological and syntactic relations are usually local in nature. Dickinson's results show that the "distrust the fringe" heuristic works better, so that is the one we employ. This means that we effectively keep all variation trigrams with the variation nucleus being the middle word.

After obtaining the set of types with an error in their part-of-speech annotation, we turn our attention to the issue of automatically correcting them. To that end, we have to determine which tokens are the erroneous ones, and decide what the correct part-of-speech tag for each of them is. We use a part-of-speech tagger, TreeTagger[5] (Schmid, 1994), to address both of these tasks simultaneously.

### 4.2.3 Complex Ambiguity Tags

Before training the tagger, based on the proposal of Dickinson (2005), we preprocess the corpus using the *ambiguity classes* of the individual words in conjunction with the list of potentially erroneous tokens to assign *complex ambiguity tags* to some of the words. Ambiguity class of a word is basically a set of the different classes the word may belong to; in other words, the set of the different tags with which the word appears in the corpus. The idea behind this is that words in the same ambiguity class behave in a similar way, while words in different ambiguity classes behave differently, and we want to make the tagger aware of this difference.

For example, consider the words *before* and *away*. As one can see in (4a) and (5a), both may act as adverbs (`pos=adverb`). However, *before* may also act as a preposition (`pos=adp`) as in (4b), which makes it a part of the ambiguity class `pos=adp/pos=adv`; in contrast, *away* is never a preposition, but may act as a particle of a phrasal verb as in (5b), thus being in the ambiguity class `pos=adv/pos=part`.

(4) a. Our international efforts were far greater than ever *before*.`pos=adv` .

b. If mutation and natural selection slowly but surely give rise to more and more advanced forms of life, then it was only a matter of eons *before*.`pos=adp` splendid beings endowed with reason, self-awareness and taste shimmered onto the scene .

(5) a. " One dealer told me that if he had more cars, he 'd sell them right *away*.`pos=adv` , " says Takuro Endo , Nissan executive vice

[5] `http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/`

president .

b. " I 've never met a ghost that could n't be explained *away*.pos=part by perfectly natural means , " he says .

Complex ambiguity tag consists of the ambiguity class of the word, and of the tag the word actually has. So the complex ambiguity tag of *before* would be <pos=adp/pos=adv,pos=adv> in (4a) and <pos=adp/pos=adv,pos=adp in (4b); and the complex ambiguity tag of *away* would be <pos=adv/pos=part,pos=adv> in (5a), and <pos=adv/pos=part,pos=part> in (5b). However, we want to assign a complex ambiguity tags only to the words that are relevant to the correction. To do just that, we have implemented the procedure described by Dickinson (2005) as follows:

1. *Every word which is a variation word (i.e. nucleus of a non-fringe variation) or type-identical to a variation word is assigned:*
   (a) *a complex tag reflecting the ambiguity class of all relevant ambiguities [...] in the 3-grams.*
   (b) *a simple tag reflecting no ambiguity, if tag is irrelevant.*
2. *Based on their unigram tags, non-variation words are assigned:*
   (a) *a complex tag, if and only if this word's ambiguity tag also appears as a variation ambiguity.*
   (b) *a simple tag otherwise.*

For a given type, the ambiguity (i.e. the tag) is relevant if it occurs with at least 10 % or at least 10 tokens of the type. (Just one of the two conditions is enough.)

### 4.2.4 Tagging

For training, the TreeTagger requires some training data, a lexicon, and a tagset. In our case, the whole corpus is used as the training data, and the lexicon and tagset are generated from the corpus. The lexicon contains all types from the corpus, each with the list of all tags with which they appear, except for the ordinal and cardinal numbers consisting of digits. The tagset contains all possible tags; in our case, we can also generate it from the corpus.

The first attempt to train the TreeTagger was unsuccessful on some of the corpora. We have empirically determined that some of the tags we use exceed the TreeTagger's limit on the length of the tags, which is not really surprising given the fact that some of the tags we use are unusually long and the complex ambiguity tags created from them are yet several times longer. To bypass this problem, we have converted every tag to a number (in the order in which they appeared in the corpus) and used these numbers as our tags (including in the lexicon and the tagset) while working with the tagger.

After that, we have trained the tagger on all the corpora[6] with default parameters except for the "end-of-sentence part-of-speech tag" option `-st`; we have

[6]Except on [fa], [ja], [ro] and [ru], where we encountered some other technical problems.

used `-st pos=punc`, as that is the most common end-of-sentence punctuation tag.[7]

### 4.2.5 Correction

After training the tagger on the whole corpus in question, we run it on the same corpus. We feed the output together with the set of variation $n$-grams into the Treex block `HamleDT::Util::CorrectPOSInconsistencies` we have implemented. The block goes through the whole corpus, and when it encounters a variation $n$-gram, it compares the original corpus annotation with the one provided by the tagger, and prints[8] the result of the comparison. Three cases can be distinguished:

- The tag assigned to the token by the tagger is the same as the one in the corpus. We consider the token to be in fact annotated correctly and keep the original tag.
- The tag assigned to the token by the tagger is different than the one in the corpus, and this type occurs with such a tag somewhere else in the corpus. We consider the token to be annotated erroneously, and propose to change its annotation to the one assigned to it by the tagger.
- The tag assigned to the token by the tagger is different than the one in the corpus, but this type does not occur with such a tag at all. We disregard this token, as we cannot determine whether the error is in the corpus, or in the tagger, and keep the original tag.

The following examples show each of the three cases:

(6) Those employees are suspected of illegally gaining an *estimated* $ 376.8 million , the prosecutor was quoted as saying by the Excelsior news service .

(7) We had intercepted during the year an *estimated* $ 5 billion street value of cocaine .

(8) The jokes are n't just on the *Japanese* , though .

In (6), the variation nucleus *estimated* (a part of the variation trigram[9] *an estimated $*) has the tag `pos=adj|degree=pos`[10] in the corpus, and the tagger assigns it the same tag, so we make no change. In (7), the variation nucleus *estimated* has the tag `pos=verb|verbform=part|tense=past|aspect=perf`[11] in the corpus, but the tagger assigns it the tag `pos=adj|degree=pos`; *estimated* appears elsewhere in the corpus in the same context with this tag, so we accept it as the new, corrected tag for the word. In (8), the variation nucleus *Japanese* has the tag `pos=noun|nountype=prop|number=sing`, and the tagger assigns the

[7] Therefore we have not encoded the `pos=punc` tag into a number, but left it as it was.

[8] It would be trivial to change the block to instead – or in addition – to modify the corpus, but we decided to first evaluate the results.

[9] or maybe even a longer $n$-gram, but we do not check for that

[10] the positive degree of an adjective

[11] perfective past participle of a verb

tag `pos=adj|degree=pos`; however, because the word *Japanese* never appears with this tag in the given context, we reject the tag by the tagger and keep the one that is in the corpus.

## 4.3 Dependencies

In the case of dependency annotation, an annotation instance is the oriented labeled dependency edge between a pair of words. A virtual edge with a technical label NIL is added between the words that are not connected by an edge in the corpus to allow for the recognition of attachment errors. We work with the harmonized treebanks, so we use the HamleDT annotation style. It might be beneficial to reduce the number of indirectly encoded relations by using a different structure for e.g. prepositional phrases, clauses introduced by a subordinating conjunction, and coordinations, but it would have to be changed in the harmonization step – due to possibly complex interactions of the structures mentioned, their conversion to encode them directly would not be completely reversible, so the reverse conversion back to the HamleDT style would most likely introduce some new errors.

### 4.3.1 Algorithm

For the detection of inconsistencies in dependency annotation, we use the script `decca-dep.py` of the DECCA project, which implements the algorithm described by Boyd et al. (2008) as follows:

1. *Compute the set of nuclei:*
   (a) *Find all dependency pairs, store them with their category label. The dependency relations annotated in the corpus are handled as nuclei of size two and mapped to their label plus a marker of the head (`L`/`R`). The labels of overlapping type-identical nuclei are collapsed into a set of labels.*
   (b) *For each distinct type of string stored as a dependency, search for non-dependency occurrences of that string and add the nuclei found with the special label NIL.*
   *To obtain an algorithm efficient enough to deal with large corpora, we adopt the following measures from [Dickinson and Meurers (2005a)]:*
      - *A trie data structure is used to store all potential nuclei and to guide the search for NIL nuclei.*
      - *The search is limited to pairs occurring within the same sentence.*
      - *NIL nuclei which would be type-identical to and overlap with a genuine dependency relation in the same sentence are not considered.*
2. *Compute the set of variation nuclei by determining which of the stored nuclei have more than one label.*

We have slightly modified the implementation: We have refactored the command-line options parsing and added command-line support for a few more options previously requiring a modification in the code; we also provided code for textual output to simplify the following work on correction.

### 4.3.2 Heuristics

Boyd et al. (2008) offer three different heuristics:

- the *non-fringe* heuristic requires a word of identical context on both sides of both words composing the variation nucleus,
- the *NIL internal context* heuristic requires the "NIL nuclei to have the same internal context as an annotated dependency", with no requirements on the annotated dependencies, and finally
- the *dependency context* heuristic requires the head of the variation nucleus to have the same dependency label.

Their results support the intuition that the strictest of them, the non-fringe heuristic, provides the highest precision, so that is the one we have decided to use. It would be possible to combine it with the dependency context heuristic. However, their experiments show only a marginal improvement in precision against a drop in recall of about 50 %; we expect a similar effect in our experiments, and while we focus mostly on precision, we do not consider such offset worth it.

After obtaining the list of suspected errors, we attempt to automatically correct them, using a similar method as for the part-of-speech annotation: we train a non-projective dependency parser, the MSTParser[12] (McDonald et al., 2005) and use its output to determine the correct structure and label for each variation nucleus.

### 4.3.3 Parsing

For each treebank,[13] we train the MSTParser on all the data with the default parameters except for `decode-type`, which we set to `non-proj` to make the parser use the non-projective parsing algorithm.

### 4.3.4 Correction

Using the model trained on the whole corpus, we then run the parser on the whole corpus as well. The correction step itself is then basically the same as in the case of part-of-speech annotation. We have implemented a Treex block `HamleDT::Util::CorrectDependencyInconsistencies.pm`, which takes the list of variation ngrams and the the parser output, and for each variation nucleus that passed the heuristic, it compares its annotation in the corpus with the annotation proposed by the parser, which again leads to three possible cases:

[12] `http://www.seas.upenn.edu/~strctlrn/MSTParser/MSTParser.html`

[13] Except for [cs]; it seems there is a memory leak during the training which causes it to fail; we suspect that a cycle or some other non-permitted structure somehow emerged during the conversion to the CoNLL or MST format.

1. Both annotations match, i.e. either there is a NIL (meaning *no*) dependency, or the dependency edge has the same orientation and label.

2. The annotations do not match, but the one proposed by the parser is not a variation that appears in the corpus.

3. The annotations do not match, the one proposed by the parser appears in the corpus, and

   (a) there are two different labels of an otherwise same dependency edge, or

   (b) there is a NIL dependency in the corpus and a "real" dependency in the parser output, or vice versa.

The case 1 is a genuine ambiguity and we consider this variation token to be annotated correctly.

In the case 2, there must be error either in the corpus, or in the parser – but we have no way to determine whether it is the first, or the second, and by trying to correct the supposed error, we would actually introduce more inconsistency into the treebank, so instead we disregard such cases.

The remaining cases are errors. The case 3a is a *labeling* error, and we correct the label according to the parser. In the case 3b, we move the dependent word under the parent proposed by the parser, and assign the label in the same way; it is an *attachment* error.

# 5. Experiments and Results

We applied the methods for automatic detection and correction of errors in part-of-speech and dependency annotation, described in the previous chapter, on the corpora in HamleDT. Some of the treebanks were excluded from some or all the experiments. A priori, we decided not to include [ar] and [sk]. The former was in the process of switching to different source data (from CoNLL 2007 to Prague Arabic Dependency Treebank 1.5) and there were some issues with empty attributes for a non-negligible number of nodes. The latter apparently contains at least part of the data twice with two different annotations, which presents an insurmountable problem for our method, as many of the doubly-annotated sentences would be detected as inconsistencies. A few others were excluded a posteriori due to some technical problems; these are listed in the appropriate sections below.

In this chapter, we first present the results for the part-of-speech annotation in Section 5.1 and the results for the dependency annotation in Section 5.2, and then the evaluation of both in Section 5.3. Finally, in Section 5.4, we discuss some ways in which our method or the harmonization process might be improved.

## 5.1 Results for Part-of-speech Annotation

In this section, we first present the results for the detection and then for the correction of errors in the part-of-speech annotation.

### 5.1.1 Detection

We were unable to finish the detection of inconsistencies in [de] because of space limitations, as the incomplete list of variation $n$-grams had a size in the order of tens of GBs; we suspect that some texts are present in the treebank more than once.

The results of the detection step of the method are summarized in Table 5.1. Let us first consider the left part of the table. In the second and third column, we can see that the number of variation unigrams, i.e. variation nuclei candidates, varies from less than one hundred types / several hundreds of tokens to a few tens of thousands of types / hundreds of thousands of tokens. However, one must bear in mind that the treebanks in corpora vary greatly in their size – the smallest corpus, [te] with only 5722 tokens, is smaller than the largest one, [cs] with 1 503 738 tokens, by a factor of almost 263, so the raw numbers do not tell us much. The fifth column shows a more accessible measure: the maximal corpus coverage tells us the proportion of tokens in each treebank that exhibit variation – and these are the only tokens we are able to consider in any way while using this method. The coverage seems to correlate positively with the treebank size. Given that a word must appear at least two times in the corpus to be able to exhibit variation, one of the contributions to this correlation might be the ratio of hapax legomena to the vocabulary size – Fengxiang (2010) shows that, at least in English, this ratio decreases with increasing text size for texts

| Language | Variation unigrams | | Corpus size | Max. corpus coverage | Nonfringe variation nuclei | | Actual coverage |
|---|---|---|---|---|---|---|---|
| | types | tokens | (tokens) | (% of tokens) | types | tokens | (% of tokens) |
| [bg] | 1346 | 35 967 | 196 151 | 18.34 | 260 | 1240 | 0.63 |
| [bn] | 148 | 1234 | 7252 | 17.02 | 0 | 0 | 0.00 |
| [ca] | 2211 | 220 483 | 443 317 | 49.73 | 767 | 5329 | 1.20 |
| [cs] | 25 525 | 703 690 | 1 503 738 | 46.80 | 3483 | 18 800 | 1.25 |
| [da] | 987 | 34 338 | 100 238 | 34.26 | 27 | 92 | 0.09 |
| [el] | 1463 | 25 733 | 70 223 | 36.64 | 147 | 489 | 0.70 |
| [en] | 3751 | 206 058 | 451 576 | 45.63 | 947 | 4499 | 1.00 |
| [es] | 2499 | 203 872 | 477 810 | 42.67 | 563 | 3979 | 0.83 |
| [et] | 180 | 2770 | 9491 | 29.19 | 11 | 26 | 0.27 |
| [eu] | 2889 | 47 140 | 151 604 | 31.09 | 88 | 452 | 0.30 |
| [fa] | 2566 | 118 808 | 189 572 | 62.67 | 676 | 2927 | 1.54 |
| [fi] | 310 | 2207 | 58 576 | 3.77 | 1 | 2 | 0.00 |
| [grc] | 4094 | 164 053 | 308 882 | 53.11 | 2045 | 13 978 | 4.53 |
| [hi] | 6096 | 263 485 | 294 509 | 89.47 | 3737 | 25 261 | 8.58 |
| [hu] | 712 | 43 011 | 139 143 | 30.91 | 46 | 239 | 0.17 |
| [it] | 842 | 26 515 | 76 295 | 34.75 | 77 | 205 | 0.27 |
| [ja] | 190 | 63 453 | 157 172 | 40.37 | 79 | 355 | 0.23 |
| [la] | 1697 | 24 751 | 53 143 | 46.57 | 114 | 398 | 0.75 |
| [nl] | 1606 | 94 622 | 200 654 | 47.16 | 87 | 618 | 0.31 |
| [pt] | 2150 | 96 876 | 212 545 | 45.58 | 1028 | 4531 | 2.13 |
| [ro] | 469 | 13 092 | 36 150 | 36.22 | 13 | 34 | 0.09 |
| [ru] | 11 244 | 249 266 | 497 465 | 50.11 | 582 | 2311 | 0.46 |
| [sl] | 824 | 11 082 | 35 140 | 31.54 | 86 | 391 | 1.11 |
| [sv] | 1244 | 120 308 | 197 123 | 61.03 | 514 | 2483 | 1.26 |
| [ta] | 77 | 795 | 9581 | 8.30 | 7 | 24 | 0.25 |
| [te] | 70 | 689 | 5722 | 12.04 | 8 | 16 | 0.28 |
| [tr] | 792 | 22 583 | 69 695 | 32.40 | 229 | 1359 | 1.95 |

**Table 5.1:** Variation in part-of-speech annotation.
The second and third column show the number of detected variation unigrams, i.e. all possible variation nuclei. The numbers in the two penultimate columns represent how many suspected errors we detected in each of the corpora.

shorter than 3 000 000 words (which is about twice the size of the largest corpus in HamleDT). However, we do not explore this relationship in depth.

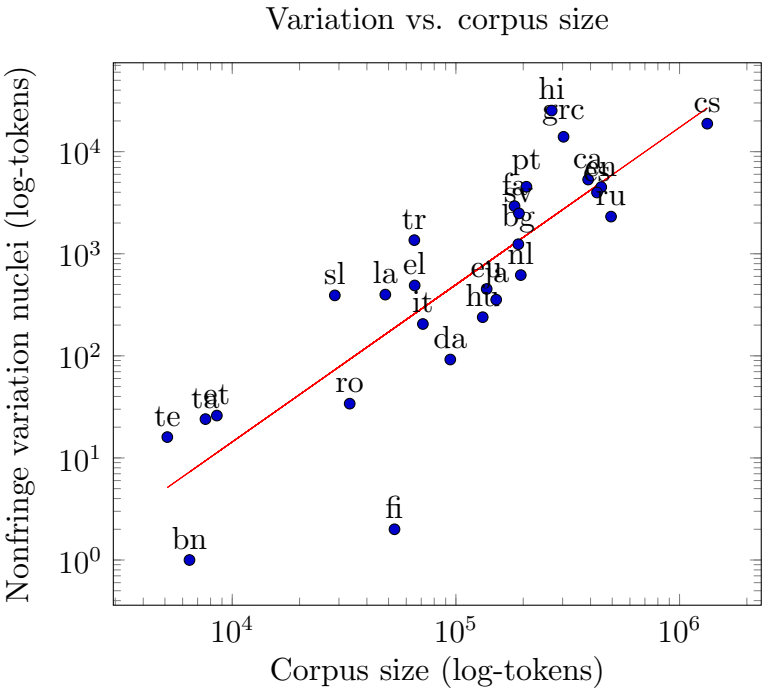



**Figure 5.1:** Nonfringe variation nuclei vs. corpus size.
(The value for [bn] was manually changed from 0 to 1 to fit in the graph.)

The last three columns are more relevant for the task at hand. The penultimate two columns show how many variation nuclei we suspect to be annotated erroneously after applying the non-fringe heuristic; it seems this heuristic cuts down the number of cases by about one or two orders of magnitude. However, bear in mind that we do not suggest that all the variation nuclei tokens are annotated incorrectly; it is merely the number of tokens of variation nuclei *types* that are (in some of their instances) annotated incorrectly according to our method. We were interested in the relation between this number and the treebank size – in Figure 5.1, we plotted the logarithm of the number of non-fringe variation nuclei tokens as a function of the logarithm of the size of the corpus in tokens. While there still seems to be a strong positive correlation, some of the treebanks have notably less, or more, suspected errors than average. The most striking example of the former is [fi], which is somewhat encouraging given that, as far as we know, it is the only treebank in HamleDT with a full double annotation and thus presumably of a very high quality. (Haverinen et al., 2011)

The last column of the table then shows the proportion of the variation nuclei tokens to the size of the treebank, and this number forms the (exclusive) upper bound on the treebank improvement by our method – we cannot reach it even with a hundred percent precision in both detection and correction, because we assume at least one correctly annotated token per each variation nucleus type. We immediately see that for some of the treebanks, our method will do very little or nothing at all even in the best case – we have detected no (potential) errors in [bn], just one in [fi], and very small proportions in e.g. [da] or [hu].

On the other hand, in ten of the treebanks the tokens of variation nuclei form a percent or more of all the tokens.

### 5.1.2 Correction

| Language | Total | Proposed new tag | | |
|---|---|---|---|---|
| | (tokens) | Same | Diff. | NA |
| [bg] | 1172 | 883 | 280 | 9 |
| [bn] | 0 | 0 | 0 | 0 |
| [ca] | 4974 | 3611 | 1282 | 81 |
| [cs] | 16 017 | 8847 | 5150 | 2020 |
| [da] | 66 | 48 | 18 | 0 |
| [el] | 477 | 321 | 136 | 20 |
| [en] | 3984 | 2496 | 1419 | 69 |
| [es] | 3711 | 2708 | 985 | 18 |
| [et] | 26 | 11 | 9 | 6 |
| [eu] | 320 | 198 | 107 | 15 |
| [fi] | 2 | 1 | 1 | 0 |
| [grc] | 8765 | 5951 | 2549 | 265 |
| [hi] | 23 636 | 16 048 | 5063 | 2525 |
| [hu] | 156 | 108 | 48 | 0 |
| [it] | 182 | 91 | 81 | 10 |
| [la] | 391 | 217 | 137 | 37 |
| [nl] | 542 | 391 | 139 | 12 |
| [pt] | 4327 | 2613 | 1408 | 306 |
| [sl] | 363 | 238 | 115 | 10 |
| [sv] | 2226 | 1251 | 900 | 75 |
| [ta] | 19 | 12 | 7 | 0 |
| [te] | 14 | 6 | 6 | 2 |
| [tr] | 1068 | 586 | 439 | 43 |

**Table 5.2:** Part-of-speech annotation correction.
The second column contains the total number of tokens marked as suspected errors in the detection step. The last three columns show the number of tokens for which the newly proposed tag is 1. same as the original one, 2. different from the original one, 3. not applicable (and discarded).

After detecting the potential errors, we try to automatically correct them. As mentioned in the previous section, we did not obtain usable results for the detection on [de] and [sk], and so we could not proceed with the correction step on these treebanks. We were also unable to finish the correction step

on [fa], [ja], [ro] and [ru], because the attempts to train the tagger on these corpora were unsuccessful for various reason: in [fa], the tagger inexplicably encountered an unknown tag; punctuation in [ja] and [ro] is tagged differently, which combined with our tag encoding makes the tagger unable to recognize it; and there were apparently some words with no tags in [ru].

The results for the correction step are presented in Table 5.2. The second column is the number of tokens we considered for a correction, and the following three each stand for one of the three possible outcomes as described in Section 4.2.5 – preservation of the original tag, correction of the tag, or saying we are unable to decide (when the tagger disagrees with the original tag, but proposes a tag that is not part of the acceptable variation for the given type). We can see that in (almost) every treebank, a majority of cases result in the preservation of the original tag. The proposed inapplicable tags make up an unsubstantial proportion in most of the treebanks, and while they might provide some insights, we decided to leave this issue for a prospective further work.

## 5.2 Results for Dependency Annotation

In this section, we first present the results for the detection and then for the correction of errors in the dependency annotation.

### 5.2.1 Detection

We have successfully run the detection step of the method on all twenty-eight treebanks included in this experiment. The results are summarized in Table 5.3 and visualized in Figure 5.2.

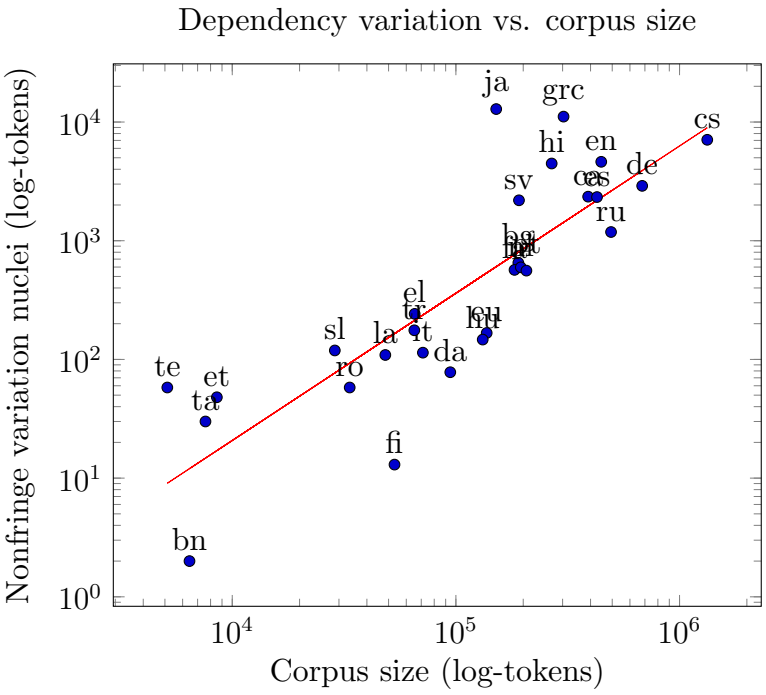



**Figure 5.2:** Dependency non-fringe variation nuclei vs. corpus size.

The table does not contain the numbers of variation nuclei candidates, because the script is implemented in such a way that the heuristic is not applied separately after obtaining the set of the variation nuclei candidates, but instead is applied immediately after obtaining this set for $n$-grams of given length. And unlike for the part-of-speech annotation, there is no straightforward way of computing the corpus coverage, because in addition to the dependencies present in the corpus, we also consider the NIL dependencies in some cases; however, we assume the corpus coverage is similar as in the part-of-speech annotation error detection, i.e. in the interval between zero and a few percent at most.

### 5.2.2 Correction

In the correction step, we had to exclude [cs], because we were unable to successfully train the parser on this corpus due to some memory-related errors; as we already mentioned in Section 4.3.3, we suspect there was some error during the format conversion, leading to a structure causing an infinite loop or a memory leak in the training of the parser.

The results for this step for the other twenty-seven treebanks are presented in Table 5.4. As we can see, the numbers of tokens considered for the correction in each treebank on the level of dependencies are in the same general vicinity as on the level of part-of-speech annotation; for some treebanks, the former number is the higher, for others, the latter one. The ratio between the number of the unchanged and of the changed positions is also relatively similar. One difference is in the number of proposed annotations that are not applicable, which is considerably higher in the case of dependency annotation.

## 5.3 Evaluation

The results of the method were manually evaluated for several languages; it was six languages for the part-of-speech annotation, and seven languages for the dependency annotation. These languages were chosen based on what language experts (mainly linguistically knowledgeable university students of the given languages) were available and willing to work on the evaluations. For each evaluated language, two random samples of one hundred variation nuclei tokens each were taken – one from the variation nuclei that were not changed (i.e. we say they were annotated correctly), and one from the variation nuclei that were changed (we say they were annotated erroneously, and propose a correction).[1] Every token was given with its context (the whole sentence in which it occurred), the original annotation, and the proposed new annotation. The evaluators went through both samples, assessing the correctness of both the original and the new annotation.[2]

In the case of the unchanged tokens, there are just two possibilities: either the original annotation is correct, which means we have made a right decision in not changing it; or the original annotation is incorrect, and we have thus

[1] For some of the treebanks, there were less than one hundred changed and/or unchanged positions; in that case, we included all cases in the evaluation.

[2] In the sample of the unchanged tokens, the original and the new annotations are of course the same.

| Language | Corpus size | Nonfringe Variation nuclei | |
|---|---|---|---|
| | (tokens) | types | tokens |
| [bg] | 196 151 | 261 | 652 |
| [bn] | 7252 | 1 | 2 |
| [ca] | 443 317 | 614 | 2351 |
| [cs] | 1 503 738 | 1704 | 7085 |
| [da] | 100 238 | 35 | 78 |
| [de] | 680 710 | 851 | 2899 |
| [el] | 70 223 | 102 | 242 |
| [en] | 451 576 | 959 | 4622 |
| [es] | 477 810 | 515 | 2333 |
| [et] | 9491 | 24 | 48 |
| [eu] | 151 604 | 70 | 167 |
| [fa] | 189 572 | 225 | 568 |
| [fi] | 58 576 | 6 | 13 |
| [grc] | 308 882 | 2785 | 11 108 |
| [hi] | 294 509 | 1094 | 4473 |
| [hu] | 139 143 | 66 | 147 |
| [it] | 76 295 | 51 | 114 |
| [ja] | 157 172 | 689 | 12 860 |
| [la] | 53 143 | 37 | 109 |
| [nl] | 200 654 | 120 | 597 |
| [pt] | 212 545 | 187 | 560 |
| [ro] | 36 150 | 18 | 58 |
| [ru] | 497 465 | 356 | 1184 |
| [sl] | 35 140 | 38 | 119 |
| [sv] | 197 123 | 458 | 2191 |
| [ta] | 9581 | 14 | 30 |
| [te] | 5722 | 28 | 58 |
| [tr] | 69 695 | 73 | 176 |

**Table 5.3:** Variation in dependency annotation.

made a mistake and retained an error in the corpus. In the case of the changed tokens, the situation is more complicated with four possibilities, as both the original and the new annotation may be correct or incorrect independently of each other. Finally, there is one extra possibility common for both cases: it might be impossible (either for the specific annotator, or in principle) to determine the correct annotation. For an overview of all the possibilities, see

| Language | Total | Proposed new annotation | | |
|---|---|---|---|---|
| | (tokens) | Same | Diff. | NA |
| [bg] | 662 | 302 | 191 | 169 |
| [bn] | 0 | 0 | 0 | 0 |
| [ca] | 2353 | 1348 | 373 | 632 |
| [da] | 78 | 37 | 17 | 24 |
| [de] | 2859 | 1248 | 420 | 1191 |
| [el] | 246 | 109 | 56 | 81 |
| [en] | 4637 | 2020 | 803 | 1814 |
| [es] | 2341 | 1188 | 353 | 800 |
| [et] | 48 | 26 | 10 | 12 |
| [eu] | 166 | 67 | 36 | 63 |
| [fa] | 586 | 281 | 149 | 156 |
| [fi] | 13 | 5 | 2 | 6 |
| [grc] | 11 052 | 4151 | 1742 | 5159 |
| [hi] | 4482 | 2397 | 828 | 1257 |
| [hu] | 144 | 56 | 26 | 62 |
| [it] | 114 | 54 | 32 | 28 |
| [ja] | 12 799 | 9360 | 1127 | 2312 |
| [la] | 108 | 61 | 20 | 27 |
| [nl] | 597 | 375 | 130 | 92 |
| [pt] | 560 | 309 | 115 | 136 |
| [ro] | 58 | 35 | 15 | 8 |
| [ru] | 1170 | 635 | 229 | 306 |
| [sl] | 121 | 52 | 26 | 43 |
| [sv] | 2189 | 1035 | 353 | 801 |
| [ta] | 30 | 12 | 8 | 10 |
| [te] | 57 | 33 | 18 | 6 |
| [tr] | 180 | 82 | 45 | 53 |

**Table 5.4:** Dependency annotation correction.

Table 5.5.[3]

For each evaluated treebank, we then went through all the samples, noting how many times each of the possibilities appeared, and then used these numbers to estimate the separate precision for the changed and the unchanged positions for both the detection and the correction step. The estimated precision for each

[3]For the sake of completeness, we include all combinations of true/false values of the three variables "annotation changed", "original annotation correct", and "new annotation correct", even though two of them are logically impossible, because if the annotation was unchanged, the latter two variables must have the same value.

| Case | Data | | | Meaning | |
|---|---|---|---|---|---|
| | Annotation changed? | Original annotation correct? | New annotation correct? | Detected correctly? | Corrected correctly? |
| 1 | no | yes | yes | no | yes |
| 2 | no | yes | no | —* | —* |
| 3 | no | no | yes | —* | —* |
| 4 | no | no | no | yes | no |
| 5 | yes | yes | yes | no | yes |
| 6 | yes | yes | no | no | no |
| 7 | yes | no | yes | yes | yes |
| 8 | yes | no | no | yes | no |
| 9 | either | ? | ? | ? | ? |

* not possible

**Table 5.5:** Possible cases in evaluation.

of these four cases is calculated as follows ("Case #" refers to the appropriate line of the Table 5.5):

- detection on unchanged positions:

$$\frac{\text{correctly detected unchanged positions}}{\text{total number of unchanged positions}} = \frac{\text{Case 4}}{\text{Case 4} + \text{Case 1}}$$

- detection on changed positions:

$$\frac{\text{correctly detected changed positions}}{\text{total number of changed positions}} = \frac{\text{Case 7} + \text{Case 8}}{(\text{Case 7} + \text{Case 8}) + (\text{Case 5} + \text{Case 6})}$$

- correction on unchanged positions:

$$\frac{\text{correctly corrected unchanged positions}}{\text{total number of unchanged positions}} = \frac{\text{Case 1}}{\text{Case 1} + \text{Case 4}}$$

- correction on changed positions:

$$\frac{\text{correctly corrected changed positions}}{\text{total number of changed positions}} = \frac{\text{Case 5} + \text{Case 7}}{(\text{Case 5} + \text{Case 7}) + (\text{Case 6} + \text{Case 8})}$$

We actually report a range for each of the precisions, where the lower bound is obtained by grouping the undetermined tokens (Case 9) with the incorrect cases (i.e. adding it to the numerator in the formulas above), and the higher bound by grouping them with the correct cases (i.e. adding it to both the denominator and the numerator). The estimated total detection precision[4] and

[4]Note that we differ from Dickinson (2005) in the definition of detection precision – we count the proportion of detected *tokens* that are true errors, while Dickinson counts the proportion of detected *types* with at least one token annotated incorrectly.

total correction precision (again for each treebank separately) are then calculated as an average of the precision for changed/unchanged positions weighted by the total number of changed/unchanged positions in the treebank.

We also compute so-called *corpus precision*, which is a baseline correction precision based on the ratio of the detected tokens that would be annotated correctly if we made no changes in the corpus over all the detected tokens, and *expected corpus improvement*, which is the percentage of the tokens in the treebank that, based on the estimated precision, we expect to improve, i.e. that were originally annotated erroneously, but are annotated correctly after applying our method[5].

### 5.3.1 Part-of-speech Tags

The languages used for the evaluation of the method for the detection and correction of errors in part-of-speech annotation are [en], [es],[6] [hu], [nl], [sv] and [tr]. The evaluators worked on their own, because the Interset is mostly intuitive and well-documented; in case of doubt, they were instructed to make an informal note about the morphological properties of the word in question. We then went through the samples, deciding on the disputable cases based on the notes from the evaluators, consulting with them when necessary, and estimated the precision as described above.

| Lang. | Detection precision (%) | | |
|---|---|---|---|
| | Changed | Unchanged | Total |
| [en] | 57–58 | 81–83 | 72–74 |
| [es] | 63 | 81 | 68 |
| [hu] | 76 | 87–89 | 83–85 |
| [nl] | 26–29 | 89–91 | 72–75 |
| [sv] | 81 | 48 | 62 |
| [tr] | 41–48 | 59–62 | 52–56 |

**Table 5.6:** Evaluation of part-of-speech tags error detection.

Table 5.6 shows the estimated precision for the detection of errors in the part-of-speech annotation on both changed and unchanged position, as well as total precision. As has been mentioned, we cannot directly compare our detection precision to the detection precision of Dickinson (2005) because they are different measures; Dickinson achieved the precision of erroneous *types* detection of 97.64 % on WSJ and of 52.00 % on BNC Sampler corpus of English. We *can* directly compare our detection precision to the results of other methods for automatic detection of part-of-speech annotation errors, like van Halteren's (2000) method using a tagger with 20.5 % on the BNC Sampler, Eskin's (2000) anomaly detection on WSJ with 44 % or the use of five different taggers and

[5] i.e. (total correction precision − corpus precision) × $\frac{\text{number of detected (suspected) errors}}{\text{number of tokens in treebank}}$

[6] The evaluator of [es] unfortunately evaluated only the first 36 instances in both files.

shallow parsing of Loftsson (2009) on the Icelandic Frequency Dictionary with 16.6 % and 30.1 %. We see that the detection method we use works substantially better in regards to precision than other methods found in literature, and is thus well-suited as a basis for a subsequent correction, even though there is a good deal of variation in the results for different treebanks. We discuss the possible reasons of this variation in Section 5.4.

| Lang. | Correction precision (%) | | | Corpus | Expected corpus |
|---|---|---|---|---|---|
| | Changed | Unchanged | Total | precision (%) | improvement (%) |
| [en] | 63–64 | 81–83 | 74–76 | 67–68 | 0.07–0.08 |
| [es] | 34 | 81 | 47 | 16 | 0.24 |
| [hu] | 78 | 87–89 | 84–86 | 64–65 | 0.02 |
| [nl] | 69–72 | 89–91 | 84–86 | 84–85 | 0.00–0.01 |
| [sv] | 49 | 48 | 48 | 36 | 0.13 |
| [tr] | 29–36 | 59–62 | 46–51 | 35 | 0.24 |

**Table 5.7:** Evaluation of part-of-speech tags error correction.

Table 5.7 shows the estimated precision for the correction of errors in the part-of-speech annotation, as well as the baseline corpus precision and the estimated rate of how the correction would improve the treebank. Our correction precision is considerably higher for the unchanged positions than for the changed positions (with an exception of [sv], where they are about the same). One thing to note is that on all evaluated treebanks, our method beats (or at worst matches) the baseline, and in most cases by a considerable margin. In other words, we expect to almost always make the treebank better, and never any worse. More precisely, the expected corpus improvement for the six evaluated corpora ranges from 0 % to 0.24 %, and the absolute numbers of newly correct tokens from 0 to about 1147. The improvement is much more pronounced on treebanks with a lower corpus precision, which is not really surprising – there are more opportunities for a correct action, and less opportunities for an incorrect one.

For comparison, in the task of correction of part-of-speech tags, Dickinson (2005) achieved the precision of up to 86.21/85.92/86.00 %[7] on the WSJ corpus with the baseline of 76.7 % and up to 50.00/92.11/87.33 % on the BNC Sampler with the baseline of 88.67 %. For most of the evaluated treebanks, our results are somewhat comparable on the unchanged positions, and for at least some of them ([hu] and [nl]), we achieve more or less the same total correction precision. On the changed positions, our results are mostly better than Dickinson's on the BNC Sampler, but substantially worse than on WSJ. Overall, assuming similar values of recall (which we do not even try to estimate), this would mean we expect to correct more errors in most of the treebanks, as we work on treebanks with a lower corpus precision.

[7]Henceforward, when reporting precision, the values in a slash-delimited triple refer to the values for changed positions, unchanged positions, and all positions, respectively.

### 5.3.2 Dependencies

The languages we used for the evaluation of the method for the detection and correction of errors in dependency annotation are [en], [es], [hu], [nl], [ru], [sv] and [tr]. None of the evaluators are experts in the HamleDT annotation style nor the analytic layer of the PDT, so we worked in tandem with each of the evaluators – they provided the knowledge of the language, and we assessed the annotation based on their explanation of the sentence structure and meaning. After that, we have again estimated the precision as described above.

| Lang. | Detection precision (%) | | |
|---|---|---|---|
| | Changed | Unchanged | Total |
| [en] | 51–58 | 67–72 | 62–68 |
| [es] | 70–72 | 36–38 | 44–46 |
| [hu] | 69–69 | 45–46 | 52–54 |
| [nl] | 34–40 | 31–32 | 32–34 |
| [ru] | 65–73 | 64–80 | 64–78 |
| [sv] | 65–67 | 65–68 | 65–68 |
| [tr] | 59–67 | 54–56 | 56–60 |

**Table 5.8:** Evaluation of dependency annotation errors detection.

Table 5.8 shows the estimated precision for the detection of errors in the dependency annotation on both changed and unchanged positions, as well as total precision. The precision values range from 32 % to 78 %, averaging approximately 53–58 %. Again, we unfortunately cannot compare our detection results to those of Boyd et al. (2008) because of the differing definition of precision.

We present a comparison with the results of some other methods for automatic detection of errors in dependency annotation: Anomaly detection of Novák and Razímová (2009) based on Apriori algorithm reached at most 20 % in their experiment on the deep-syntactic layer of PDT. Regarding the method of Volokh and Neumann (2011) using two different parsers run on the training data, we estimate its precision at up to 52 %. Even better results were reported by B. Agrawal et al. (2013) – even though they aim for high recall, their method achieves the precision of 64.58 %; however, the experiment was conducted only on inter-chunk dependency trees, and method requires a manual construction of a knowledge base of parser error patterns. The numbers suggest that the variation $n$-grams method is competitive at the least in the area of dependency annotation as well.

Table 5.9 shows the estimated precision for the correction of errors in the dependency annotation, as well as the baseline corpus precision and the expected corpus improvement. Our method beats the baseline corpus precision on five out of the seven evaluated treebanks, the exceptions being [es] and [nl]. These two treebanks were also the ones where we discovered a consistently incorrect annotation of some grammatical construction: in [es], there were many cases of preposition with an afun belonging to the prepositional phrase instead of

| Lang. | Correction precision (%) | | | Corpus | Expected corpus |
|---|---|---|---|---|---|
| | Changed | Unchanged | Total | Precision (%) | improvement (%) |
| [en] | 41–48 | 67–72 | 60–65 | 60–63 | 0.01–0.03 |
| [es]* | 23–25 | 36–38 | 33–35 | 34–36 | 0.00 |
| [hu] | 46–46 | 45–46 | 45–46 | 20–20 | 0.02 |
| [nl]† | 28–34 | 31–32 | 30–33 | 37–37 | −0.01 |
| [ru] | 54–62 | 64–80 | 61–76 | 53–64 | 0.02–0.04 |
| [sv] | 53–55 | 65–68 | 62–65 | 57–59 | 0.04–0.06 |
| [tr] | 41–49 | 54–56 | 49–54 | 18–19 | 0.06–0.07 |

* many instances of a preposition having an afun belonging to the prepositional phrase instead of AuxP

† many instances of a reversed dependency between an infinitive marker "te" and the verb it governs

**Table 5.9:** Evaluation of dependency annotation errors correction.

AuxP; in [nl], there were many instances of a reversed dependency between an infinitive marker "te" and the verb it governs. These errors are most likely caused by an inadequacy in the harmonization process, which should be rectified in the respective harmonization block; disregarding them, the baseline is beaten on the two treebanks as well. We can thus say our method works in the sense that given a harmonized treebank, we can expect either to improve it, or at least to find an erroneous pattern which should be correctable by a change in the harmonization block. The latter unfortunately requires a manual inspection of the suggested corrections; on the other hand, it is likely that the change in the harmonization block will result in a correction of a whole class of errors, including tokens not exhibiting any variation. Nevertheless, without the manual inspection, we cannot be reasonably sure the treebank can be expected to be improved by the application of the suggested corrections, and so the method is currently not suitable for a fully automatic deployment.

The estimated total precision of our method ranges from about 30 % or 45 % (excluding [es] and [nl]) to about 76 %, with the baseline varying greatly from about 18 % to about 64 % and the expected corpus improvement from −0.01 %/(+)0.01 % to 0.07 %, translating into the expected reduction in number of errors between −20/(+)26 and 134 based also on the size of the treebank. In comparison, Volokh and Neumann (2011) correct 3535 suspected errors. They assume "a very high precision" based on how their method works, however, they do not provide any numbers. Dickinson (2009) optimizes a memory-based learning approach, achieving the best result using ambiguity classes as features and constraints, obtaining the precision of 59.6 % on the changed positions and 76.7 % in total, with the baseline of 70.1 %, corresponding by our calculation to the expected treebank improvement of about 0.08 %, which is only slightly better than our results – the difference most likely being due to the inevitable, albeit slight, information loss in the harmonization.

## 5.4 Discussion

The variation $n$-grams method for the automatic detection of annotation and the subsequent automatic correction using a tagger/parser perform relatively well, as they can – with some reservations – be expected to improve the corpora on which they are applied. However, there are some issues related to HamleDT, treebanks in general, or the method itself, that may have a negative influence on the results. If we were able to rectify, circumvent, or at least mitigate these issues, the results might be improved – not to mention the possible increase in the consistency of HamleDT itself. There are also some ways in which the method could be extended. We discuss a few examples of the above in this section.

### 5.4.1 Annotation Guidelines

One of the obstacles on the way to the inter-treebank consistency is the absence of annotation guidelines for the Prague Dependencies style. The Prague Dependency Treebank has a comprehensive annotation manual[8] with detailed instructions, including boundary and problematic cases. However, this manual has been written specifically for Czech, without regard for other languages. This is not much of a problem for most of the common cases, but the detailed instructions for borderline and peripheral phenomena are usually not directly applicable for other languages and their harmonization depends a great deal on the linguistic intuition of the author of the Treex harmonization block. Some of the rules, for example for the label AuxZ ("emphasizing word"), even contain an exclusive list of particular words that can bear this label. Moreover, the Prague Dependencies style is not identical to the annotation style of PDT, and there is not even a comprehensive list of changes from one style to the other.

As a consequence, some equivalent constructions are annotated differently, based on their annotation in the original treebank and the intuition of the author of the harmonization block. It is also the reason why some of the treebanks use the new labels AuxA or Neg for articles and negative particles, respectively, while others keep using Atr or Adv for the same phenomena.

### 5.4.2 Part-of-speech Annotation

An Interset *driver* operates on a single tagset with no regard to other drivers or tagsets. The decoder simply "read[s] a string (tag) into an internal data structure, in accordance with the list of possible features and their values" (Zeman, 2008) – the intent of the decoding is to preserve the information from the original tag. However, even abstracting from the differences between languages, the tagsets still differ in their coverage of the morphological features – some features distinguishable in the language may not be distinguished in the tagset. Because the decoder does not look beyond the tag, this may lead to unnecessary inter-treebank inconsistencies. We think that the morphological annotation of at least some closed classes might benefit from a more liberal approach, taking in account the word form, for example. We do however admit that the benefit

[8] http://ufal.mff.cuni.cz/pdt2.0/doc/manuals/en/a-layer/pdf/a-man-en.pdf

might be more than offset by the increased complexity of writing the drivers, and the liberal approach might be at odds with the utility of Interset outside of HamleDT. Some semi-automatic postprocessing might thus be a better idea.

For an example inconsistency, we can look at how a dot is tagged. The symbol "." bears 16 different tags[9] across 27 treebanks, 3 of them correct:

- 19 `pos=punc`
- 10 `pos=punc|punctype=per`
- 1 `pos=punc|punctype=peri;qest;excl`

and the rest incorrect (in some of them at least the `pos` attribute has the correct value of `punc`, in others not even that):

- 3 empty tag
- 2 `pos=punc|punctype=comma`
- 1 `pos=punc|punctype=quot`
- 1 `pos=punc|punctype=excl`
- 1 `pos=punc|punctype=dash`
- 1 `pos=punc|verbform=inf`
- 1 `pos=num`
- 1 `pos=noun`
- 1 `pos=noun|animateness=inan`
- 1 `pos=noun|animateness=anim`
- 1 `hyph=hyph`
- 1 `pos=verb|number=sing|person=1|voice=act`

### 5.4.3 Dependency Annotation

In the evaluation of the correction of dependency annotation in [es], we have encountered a large number of incorrectly annotated prepositional phrases which our method was unable to correct. As described in Section 3.1.2, the preposition should be a head, bearing the auxiliary label AuxP, and its child should have the label that expresses the relation of the whole prepositional phrase to its parent. Figure 5.3 shows a simplified example of the erroneous annotation from the corpus[10]; we present the correct annotation in Figure 5.4.

In the evaluated sample, the parser never assigns the correct AuxP label to the preposition; when suggesting a different label, it is always the one that should belong to the prepositional phrase, i.e. to the child of the preposition. Therefore it seems there are some cases where the harmonization of [es] fails to

[9] in our sense, i.e. Interset representations

[10] The relevant part of the sentence in the corpus contains a coordination structure, *el emperador y la presunta divinidad de Japón* ("the emperor and the presumed divinity of Japan") with the prepositional phrase *de Japón* ("of Japan") technically depending on the conjunction *y* ("and"), modifying both *el emperador* ("the emperor") and *la presunta divinidad* ("the presumed divinity"). This should however have no effect on the labels the prepositional phrase receives.

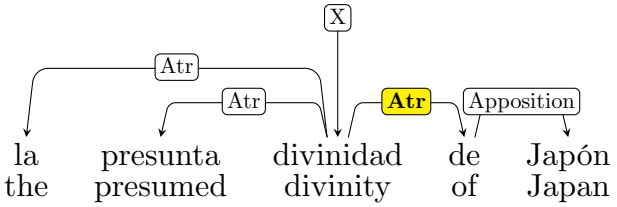



**Figure 5.3:** Example of a prepositional phrase in [es] where the preposition bears the label of the whole phrase instead of AuxP.

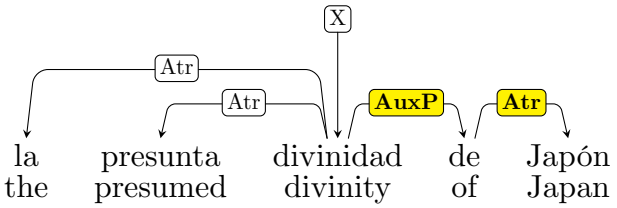



**Figure 5.4:** Correct annotation of a prepositional phrase in [es].

produce the expected result despite the correct annotation of the prepositional phrase in the original treebank.

Another treebank where we encountered some underlying problems was [nl]. In this case, it was actually two classes of problems: 1. a reversed dependency between an infinitive marker *te* and the verb it governs, 2. many instances of an ambiguity between Pnom and Sb in *wh*-questions.

We believe the infinitive marker *te* should be governed by the infinitive (as it is in [en]), and labeled either AuxV (an auxiliary verb) or AuxY (adverb or particle not classed elsewhere). However, that is not the case in the treebank; the infinitive marker is consistently annotated as governing the verb with the label AuxC (subordinate conjunction). An example from the corpus is shown in Figure 5.5.

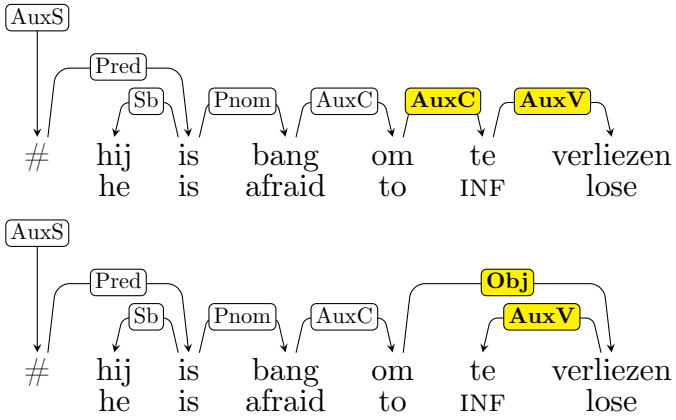



**Figure 5.5:** Example of an incorrect annotation of an infinitive construction in [nl] in comparison with the correct one.

There is quite a large number of *wh*-questions in [nl], where the verb has two arguments: Sb and Pnom. The problem lies in the fact that (at least from a Czech-centric view) the assignment of the labels to the verb arguments is

determined on a purely semantic basis, depending on the relative specificity of the entity substituted by the *wh*-word and of the other argument – for an example of both eventualities, see Figure 5.6. On the syntactic level, both cases are identical. The parser is therefore unable to generalize this structure well, let alone distinguish the two cases.

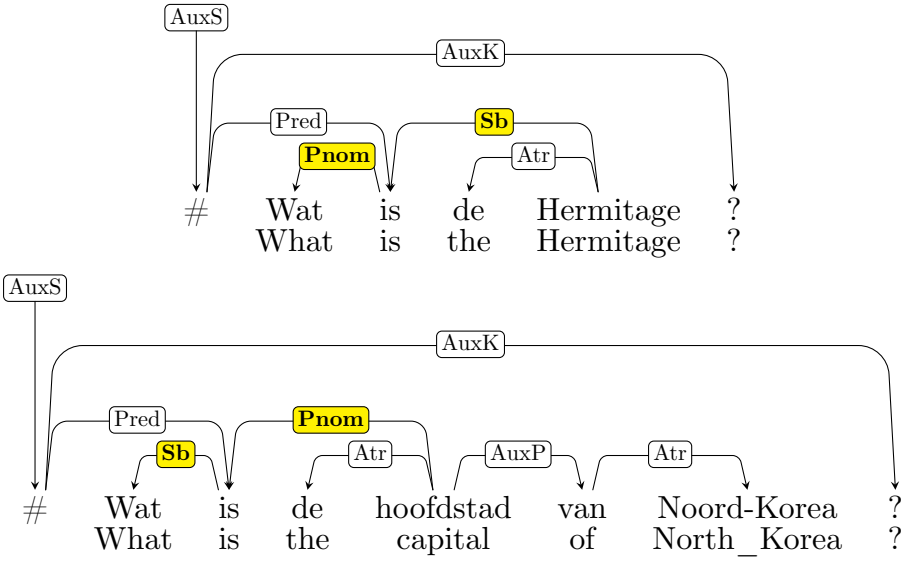



**Figure 5.6:** Example of Sb–Pnom ambiguity in *wh*-questions in [nl].

### 5.4.4 Method Extensions

The variation $n$-grams method might be improved in two basic ways. The first one is to generalize the definition of the variation nucleus, and the second one is to generalize the notion of context and use some different heuristics. It is possible to look beyond the word forms and use for example the part-of-speech tags for context or even for variation nuclei. The generalization should lead to the increase in recall, although most likely accompanied by a drop in precision. Boyd et al. (2007) Dickinson and Meurers (2005b) Dickinson and Meurers (2005a) experiment with POS tags as context and nuclei on the phrase-structure annotation, reporting a substantial increase of recall at the cost of slightly decreased precision; however, the new heuristics they devised are not easily transferable to either part-of-speech or dependency annotation.

An interesting option would be to use more than one method for detection – for example Volokh and Neumann (2011) report that the set of errors detected by their approach using two different parsers does not much overlap with the set of errors detected by the variation $n$-grams method.

The precision of our correction method would likely increase if we used the output of two taggers/parsers instead of just one; we are however unable to estimate by how much, and how large would the recall trade-off be.

As noted by Boyd et al. (2008), there are several difficulties in the data-driven approach to error correction: in the cases of variation where there is a single correct label, one cannot simply correct to the majority label, because "a

large number of variations are ties between two different labels" and even "where there is a majority tag [...], a non-majority label is actually correct"; and then there are cases of variation that is legitimate and should not be changed, which is however often not distinguishable without using non-local information.

# 6. Conclusion

In this thesis, we presented a method for the automatic detection and correction of errors in the morphological and the syntactic annotation of dependency treebanks.

First, we studied the thirty dependency treebanks of typologically diverse languages included in HamleDT, and increased the inter-treebank consistency by modifying the harmonization blocks for some of the treebanks to replace the obsolete or language-specific labels with their language-universal functional equivalents.

Then we used the variation $n$-grams method to automatically detect errors in the morphological and the syntactic annotation of the treebanks. After that, we ran a tagger/parser on each of the treebanks, and used their output to correct the errors detected in the previous step.

Both the detection and the correction were manually evaluated on a randomly selected sample of the suspected errors from several treebanks. The results of the evaluation indicate that the variation $n$-grams method is suitable for the automatic error detection on a variety of languages. Both steps have a higher precision on the morphological level, but we can expect a decrease in the number of errors as a result of applying the corrections on the syntactic level as well.

In addition to the suggested corrections, we now have a highly automated pipeline which can facilitate further experiments on the automatic detection and/or correction of errors. The acquired data can also serve as a basis for improvement of the harmonization blocks.

# Bibliography


Aduriz, Itzair, María Jesús Aranzabe, Jose Mari Arriola, Aitziber Atutxa, Arantza Díaz de Ilarraza, Aitzpea Garmendia and Maite Oronoz. 2003. Construction of a Basque dependency treebank. In. *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories*. 2003.

Afonso, Susana, Eckhard Bick, Renato Haber and Diana Santos. 2002. "Floresta sintá(c)tica": a treebank for Portuguese. In. *Proc. of LREC*. 2002, pp. 1968–1703.

Agrawal, Bhasha, Rahul Agarwal, Samar Husain and Dipti M. Sharma. 2013. An Automatic Approach to Treebank Error Detection Using a Dependency Parser. In. *Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing - Volume Part I*. Samos, Greece: Springer-Verlag, 2013, pp. 294–303. CICLing'13. Available also from WWW: ⟨`http://dx.doi.org/10.1007/978-3-642-37247-6_24`⟩. ISBN 978-3-642-37246-9.

Agrawal, Rakesh and Ramakrishnan Srikant. 1994. Fast Algorithms for Mining Association Rules in Large Databases. In. *20th International Conference on Very Large Data Bases*. 1994, pp. 478–499.

Aranzabe, M., J. Arriola, A. Atutxa, I. Balza and L. Uria. 2003. *Guía para la anotación sintáctica manual de Eus3LB (corpus del euskera anotado a nivel sintáctico, semántico y pragmático)*. 2003.

Atalay, Nart B., Kemal Oflazer, Bilge Say and Informatics Inst. 2003. The Annotation Process in the Turkish Treebank. In. *Proc. of the 4th Intern. Workshop on Linguistically Interpreteted Corpora (LINC)*. 2003.

Bamman, David and Gregory Crane. 2011. The Ancient Greek and Latin Dependency Treebanks. In Sporleder, Caroline, Antal Bosch and Kalliopi Zervanou (ed.). *Language Technology for Cultural Heritage*. 2011, pp. 79–98. Theory and Applications of Natural Language Processing. ISBN 978-3-642-20227-8.

Barnett, Vic and Toby Lewis. 1994. *Outliers in statistical data*. 3rd. Chichester: John Wiley & Sons, 1994. Wiley Series in Probability & Statistics. ISBN 0-471-93094-6.

Beek, Leonoor van der, Gosse Bouma, Jan Daciuk, Tanja Gaustad, Robert Malouf, Gertjan van Noord, Robbert Prins and Begoña Villada. 2002. Chapter 5. The Alpino Dependency Treebank. In. *Algorithms for Linguistic Processing NWO PIONIER Progress Report*. 2002. Available also from WWW: ⟨`http://odur.let.rug.nl/~vannoord/trees/Papers/report_ch5.pdf`⟩.

Bejček, Eduard, Eva Hajičová, Jan Hajič, Pavlína Jínová, Václava Kettnerová, Veronika Kolářová, Marie Mikulová, Jiří Mírovský, Anna Nedoluzhko, Jarmila Panevová, Lucie Poláková, Magda Ševčíková, Jan Štěpánek and Šárka Zikánová. 2013. *Prague Dependency Treebank 3.0*. 2013. Available also from WWW: ⟨`http://hdl.handle.net/11858/00-097C-0000-0023-1AAF-3`⟩.

Bick, Eckhard, Heli Uibo and Kaili Müürisep. 2004. Arborest – a VISL-Style Treebank Derived from an Estonian Constraint Grammar Corpus. In. *Proc. of Treebanks and Linguistic Theories*. 2004. Available also from WWW: ⟨http://beta.visl.sdu.dk/pdf/Bick_Uibo_Muurisep_TLT04.pdf⟩.

Boguslavsky, Igor, Svetlana Grigorieva, Nikolai Grigoriev, Leonid Kreidlin and Nadezhda Frid. 2000. Dependency treebank for Russian: Concept, tools, types of information. In. *Proc. of the 18th conference on Computational linguistics-Volume 2*. 2000, pp. 987–991.

Boyd, Adriane, Markus Dickinson and Walt Detmar Meurers. 2007. Increasing the Recall of Corpus Annotation Error Detection. In. *Proceedings of the Sixth Workshop on Treebanks and Linguistic Theories (TLT 2007)*. 2007. Available also from WWW: ⟨http://decca.osu.edu/publications/boyd-et-al-07b.html⟩.

Boyd, Adriane, Markus Dickinson and Walt Detmar Meurers. 2008. On Detecting Errors in Dependency Treebanks. *Research on Language and Computation*. 2008, vol. 6, no. 2, pp. 113–137. Available also from WWW: ⟨http://cl.indiana.edu/~md7/papers/boyd-et-al-08.html⟩.

Brants, Sabine, Stefanie Dipper, Peter Eisenberg, Silvia Hansen, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith and Hans Uszkoreit. 2004. TIGER: Linguistic Interpretation of a German Corpus. *Journal of Language and Computation*. 2004, vol. 2, no. 4, pp. 597–620. Special Issue.

Buchholz, Sabine and Erwin Marsi. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing. In. *Proceedings of the Tenth Conference on Computational Natural Language Learning*. New York City, New York: Association for Computational Linguistics, 2006, pp. 149–164. CoNLL-X '06. Available also from WWW: ⟨http://dl.acm.org/citation.cfm?id=1596276.1596305⟩.

Călăcean, Mihaela. 2008. *Data-driven Dependency Parsing for Romanian*. 2008. Available also from WWW: ⟨http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.153.6068&rep=rep1&type=pdf⟩.

Csendes, Dóra, János Csirik, Tibor Gyimóthy and András Kocsor. 2005. The Szeged Treebank. In. *TSD*. 2005, pp. 123–131.

Daneš, František, Zdeněk Hlavsa, Miroslav Grepl, et al. 1987. *Mluvnice češtiny 3: Skladba*. Praha: Academia, nakladatelství Československé akademie věd, 1987. 748 pp. Available also from WWW: ⟨http://www.ujc.cas.cz/sys/galerie-obrazky/mluvnice-cestiny/index.html⟩. ISBN 21-029-88.

Dickinson, Markus. 2005. *Error detection and correction in annotated corpora*. 2005. Available also from WWW: ⟨http://ling.osu.edu/~dickinso/papers/diss/⟩.

Dickinson, Markus. 2009. Correcting Dependency Annotation Errors. In. *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL-09)*. 2009. Available also from WWW: ⟨http://cl.indiana.edu/~md7/papers/dickinson09.html⟩.

Dickinson, Markus and Walt Detmar Meurers. 2003a. Detecting Errors in Part-of-Speech Annotation. In. *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL-03)*. 2003, pp. 107–114. Available also from WWW: ⟨`http://ling.osu.edu/~dickinso/papers/dickinson-meurers-03.html`⟩.

Dickinson, Markus and Walt Detmar Meurers. 2003b. Detecting Inconsistencies in Treebanks. In. *Proceedings of the Second Workshop on Treebanks and Linguistic Theories (TLT 2003)*. 2003, pp. 45–56. Available also from WWW: ⟨`http://ling.osu.edu/~dm/papers/dickinson-meurers-tlt03.html`⟩.

Dickinson, Markus and Walt Detmar Meurers. 2005a. Detecting Errors in Discontinuous Structural Annotation. In. *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-05)*. 2005. Available also from WWW: ⟨`http://ling.osu.edu/~dm/papers/dickinson-meurers-05.html`⟩.

Dickinson, Markus and Walt Detmar Meurers. 2005b. Prune Diseased Branches to Get Healthy Trees! How to Find Erroneous Local Trees in a Treebank and Why It Matters. In. *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories (TLT 2005)*. 2005, pp. 41–52. Available also from WWW: ⟨`http://cl.indiana.edu/~md7/papers/dickinson-meurers-tlt05.html`⟩.

Džeroski, Sašo, Tomaž Erjavec, Nina Ledinek, Petr Pajas, Zdeněk Žabokrtský and Andreja Žele. 2006. Towards a Slovene Dependency Treebank. In. *Proc. of LREC 2006*. Genova, Italy: European Language Resources Association (ELRA), 2006, pp. 1388–1391. Available also from WWW: ⟨`http://hnk.ffzg.hr/bibl/lrec2006/summaries/133.html`⟩.

Eskin, Eleazar. 2000. Detecting Errors Within a Corpus Using Anomaly Detection. In. *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*. Seattle, Washington: Association for Computational Linguistics, 2000, pp. 148–153. NAACL 2000. Available also from WWW: ⟨`http://dl.acm.org/citation.cfm?id=974305.974325`⟩.

Fengxiang, Fan. 2010. Squibs: An Asymptotic Model for the English Hapax/Vocabulary Ratio. *Computational Linguistics, Volume 36, Issue 4 - December 2010*. 2010. Available also from WWW: ⟨`http://aclweb.org/anthology/J10-4003`⟩.

Firth, John Rupert. 1957. A synopsis of linguistic theory 1930–1955. In. *Studies in Linguistic Analysis*. Oxford: Blackwell, 1957, pp. 1–32.

Hajič, Jan. 1998. Building a Syntactically Annotated Corpus: The Prague Dependency Treebank. In Hajičová, Eva (ed.). *Issues of Valency and Meaning. Studies in Honor of Jarmila Panevová*. 1998, pp. 12–19.

Hajič, Jan, Jarmila Panevová, Eva Buráňová, Zdeňka Urešová, Alevtina Bémová, Jan Štěpánek, Petr Pajas and Jiří Kárník. 2004. *UFAL/CKL technical report*. 2004.

Hajič, Jan, Jarmila Panevová, Eva Hajičová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, Marie Mikulová, Zdeněk Žabokrtský and Magda Ševčíková-Razímová. 2006. *Prague Dependency Treebank 2.0*. Philadelphia, PA, USA: Linguistic Data Consortium, 2006. CD-ROM, Linguistic Data Consortium, LDC Catalog No.: LDC2006T01, Philadelphia⟨http://hdl.handle.net/11858/00-097C-0000-0001-B098-5⟩. ISBN 1-58563-370-4.

Haverinen, Katri, 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 International Conference on Dependency Linguistics (Depling'11), Barcelona, Spain*. 2011, pp. 115–124.

Haverinen, Katri, Timo Viljanen, Veronika Laippala, Samuel Kohonen, Filip Ginter and Tapio Salakoski. 2010. Treebanking Finnish. In Dickinson, Markus, Kaili Müürisep and Marco Passarotti (ed.). *Proc. of the Ninth International Workshop on Treebanks and Linguistic Theories (TLT9)*. 2010, pp. 79–90. Available also from WWW: ⟨http://hdl.handle.net/10062/15936⟩.

Husain, Samar, Prashanth Mannem, Bharat Ambati and Phani Gadde. 2010. The ICON-2010 tools contest on Indian language dependency parsing. In. *Proc. of ICON-2010 Tools Contest on Indian Language Dependency Parsing*. 2010.

Kato, Yoshihide and Shigeki Matsubara. 2010. Correcting Errors in a Treebank Based on Synchronous Tree Substitution Grammar. In. *Proceedings of the ACL 2010 Conference Short Papers*. Uppsala, Sweden: Association for Computational Linguistics, 2010, pp. 74–79. ACLShort '10. Available also from WWW: ⟨http://dl.acm.org/citation.cfm?id=1858842.1858856⟩.

Kawata, Yasuhiro and Julia Bartels. 2000. Stylebook for the Japanese Treebank in Verbmobil. In. *Report 240*. 2000.

Kromann, Matthias T., Line Mikkelsen and Stine Kern Lynge. 2004. *Danish Dependency Treebank*. 2004. Available also from WWW: ⟨http://code.google.com/p/copenhagen-dependency-treebank/⟩.

Loftsson, Hrafn. 2009. Correcting a POS-Tagged Corpus Using Three Complementary Methods. In. *Conference of the European Chapter of the Association for Computational Linguistics*. 2009, pp. 523–531.

Marcus, Mitchell P., Beatrice Santorini and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*. 1993, vol. 19, no. 2, pp. 313–330.

Marneffe, Marie-Catherine de and Christopher D. Manning. 2008. The Stanford Typed Dependencies Representation. In. *Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*. Manchester, United Kingdom: Association for Computational Linguistics, 2008, pp. 1–8. CrossParser '08. Available also from WWW: ⟨http://dl.acm.org/citation.cfm?id=1608858.1608859⟩. ISBN 978-1-905593-50-7.

McDonald, Ryan, Fernando Pereira, Kiril Ribarov and Jan Hajič. 2005. Non-projective Dependency Parsing Using Spanning Tree Algorithms. In. *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Vancouver, British Columbia, Canada: Association for Computational Linguistics, 2005, pp. 523–530. HLT '05. Available also from WWW: ⟨`http://dx.doi.org/10.3115/1220575.1220641`⟩.

Montemagni, Simonetta, Francesco Barsotti, Marco Battista, Nicoletta Calzolari, Ornella Corazzari, Alessandro Lenci, Antonio Zampolli, Francesca Fanciulli, Maria Massetani, Remo Raffaelli, Roberto Basili, Maria Teresa Pazienza, Dario Saracino, Fabio Zanzotto, Nadia Mana, Fabio Pianesi and Rodolfo Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In Abeillé, Anne (ed.). *Building and using Parsed Corpora*. Dordrecht: Kluwer, 2003, pp. 189–210. Language and Speech series.

Nilsson, Jens, Johan Hall and Joakim Nivre. 2005. MAMBA Meets TIGER: Reconstructing a Swedish Treebank from Antiquity. In. *Proc. of the NODALIDA Special Session on Treebanks*. 2005. Available also from WWW: ⟨`http://www.msi.vxu.se/users/nivre/research/Talbanken05.html`⟩.

Nivre, Joakim, Johan Hall and Jens Nilsson. 2006. MaltParser: A data-driven parser-generator for dependency parsing. In. *In Proc. of LREC-2006*. 2006, pp. 2216–2219.

Novák, Václav and Magda Razímová. 2009. Unsupervised Detection of Annotation Inconsistencies Using Apriori Algorithm. In. *Proceedings of the Third Linguistic Annotation Workshop*. Suntec, Singapore: Association for Computational Linguistics, 2009, pp. 138–141. ACL-IJCNLP '09. Available also from WWW: ⟨`http://dl.acm.org/citation.cfm?id=1698381.1698405`⟩. ISBN 978-1-932432-52-7.

Popel, Martin, David Mareček, Jan Štěpánek, Daniel Zeman and Zdeněk Žabokrtský. 2013. Coordination Structures in Dependency Treebanks. In. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Sofija, Bulgaria: Association for Computational Linguistics, 2013, pp. 517–527. ISBN 978-1-937284-50-3.

Popel, Martin and Zdeněk Žabokrtský. 2010. TectoMT: modular NLP framework. *Advances in Natural Language Processing*. 2010, pp. 293–304.

Prokopidis, Prokopis, Elina Desipri, Maria Koutsombogera, Harris Papageorgiou and Stelios Piperidis. 2005. Theoretical and practical issues in the construction of a Greek dependency treebank. In. *Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT)*. 2005, pp. 149–160.

Ramasamy, Loganathan and Zdeněk Žabokrtský. 2012. Prague Dependency Style Treebank for Tamil. In. *Proc. of LREC 2012*. 2012.

Rasooli, Mohammad Sadegh, Amirsaeid Moloodi, Manouchehr Kouhestani and Behrouz Minaei-Bidgoli. 2011. A Syntactic Valency Lexicon for Persian Verbs: The First Steps towards Persian Dependency Treebank. In. *5th Language & Technology Conference (LTC): Human Language Technologies as a Challenge for Computer Science and Linguistics*. 2011, pp. 227–231.

Rosa, Rudolf, Jan Mašek, David Mareček, Martin Popel, Daniel Zeman and Zdeněk Žabokrtský. 2014. HamleDT 2.0: Thirty Dependency Treebanks Stanfordized. In Calzolari, Nicoletta, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard and Joseph Mariani (ed.). *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC 2014)*. Reykjavík, Iceland: European Language Resources Association, 2014, pp. 2334–2341. ISBN 978-2-9517408-8-4.

Schmid, Helmut. 1994. Probabilistic Part-of-Speech Tagging Using Decision Trees. In. *International Conference on New Methods in Language Processing*. 1994, pp. 44–49. Available also from WWW: ⟨`http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/tree-tagger1.pdf`⟩.

Simov, Kiril and Petya Osenova. 2005. Extending the Annotation of BulTreeBank: Phase 2. In. *The Fourth Workshop on Treebanks and Linguistic Theories (TLT 2005)*. 2005, pp. 173–184.

Smrž, Otakar, Viktor Bielický, Iveta Kouřilová, Jakub Kráčmar, Jan Hajič and Petr Zemánek. 2008. Prague Arabic Dependency Treebank: A Word on the Million Words. In. *Proc. of the Workshop on Arabic and Local Languages (LREC 2008)*. Marrakech, Morocco: European Language Resources Association, 2008, pp. 16–23. ISBN 2-9517408-4-0.

Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez and Joakim Nivre. 2008. The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. In. *Proc. of CoNLL*. 2008.

Šimková, Mária and Radovan Garabík. 2006. Sintaksičeskaja razmetka v Slovackom nacional'nom korpuse (Синтаксическая разметка в Словацком национальном корпусе). In. *Trudy meždunarodnoj konferencii Korpusnaja lingvistika (Труды международной конференции Корпусная лингвистика) – 2006*. Sankt-Peterburg, Russia: St. Petersburg University Press, 2006, pp. 389–394. ISBN 5-288-04181-4.

Taulé, Mariona, Maria Antònia Martí and Marta Recasens. 2008. AnCora: Multilevel Annotated Corpora for Catalan and Spanish. In. *LREC*. 2008.

van Halteren, Hans. 2000. The Detection of Inconsistency in Manually Tagged Text. In. *Proceedings of the COLING-2000 Workshop on Linguistically Interpreted Corpora*. Centre Universitaire, Luxembourg: International Committee on Computational Linguistics, 2000, pp. 48–55. Available also from WWW: ⟨`http://aclweb.org/anthology/W00-1907`⟩.

Volokh, Alexander and Günter Neumann. 2011. Automatic detection and correction of errors in dependency tree-banks. In. *49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies : proceedings of the conference ; ACL HLT 2011 ; June 19-24, 2011, Portland, Oregon, USA*. Stroudsburg: ACL, 2011, pp. 346–350. HU, 978-1-932432-88-6.

Zeman, Daniel. 2008. Reusable Tagset Conversion Using Tagset Drivers. In. *Proc. of LREC 2008*. Marrakech, Morocco: European Language Resources Association (ELRA), 2008, pp. 28–30. Available also from WWW: ⟨`http://www.lrec-conf.org/proceedings/lrec2008/summaries/66.html`⟩. ISBN 2-9517408-4-0.

Zeman, Daniel, Ondřej Dušek, David Mareček, Martin Popel, Loganathan Ramasamy, Jan Štěpánek, Zdeněk Žabokrtský and Jan Hajič. 2014. HamleDT: Harmonized Multi-Language Dependency Treebank. *Language Resources and Evaluation*. 2014. ISSN 1574-020X.

Zeman, Daniel, David Mareček, Martin Popel, Loganathan Ramasamy, Jan Štěpánek, Zdeněk Žabokrtský and Jan Hajič. 2012. HamleDT: To Parse or Not to Parse? In Calzolari Nicoletta nad Choukri, Khalid, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk and Stelios Piperidis (ed.). *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. Istanbul, Turkey: European Language Resources Association (ELRA), 2012. Available also from WWW: ⟨`http://www.lrec-conf.org/proceedings/lrec2012/pdf/429_Paper.pdf`⟩. ISBN 978-2-9517408-7-7.

# List of Tables

# List of Figures

# List of Abbreviations

GEN – genitive case
SG – singular
TGPRS – present transgressive
PDT – Prague Dependency Treebank
HamleDT – Harmonized Multi-LanguagE Dependency Treebank
DECCA – Detection of Errors and Correction in Corpus Annotation
SVM – support vector machine
WSJ – Wall Street Journal corpus
NLP – natural language processing
POS – part-of-speech
CoNLL – Conference on Natural Language Learning
RIDGES – Register in Diachronic German Science
[ar] – Arabic
[bn] – Bengali
[bg] – Bulgarian
[ca] – Catalan
[cs] – Czech
[da] – Danish
[de] – German
[el] – Greek
[en] – English
[es] – Spanish
[et] – Estonian
[eu] – Basque
[fa] – Persian
[fi] – Finnish
[grc] – Ancient Greek
[hi] – Hindi
[hu] – Hungarian
[it] – Italian
[ja] – Japanese
[la] – Latin
[nl] – Dutch
[pt] – Portuguese
[ro] – Romanian
[ru] – Russian
[sk] – Slovak
[sl] – Slovenian
[sv] – Swedish
[ta] – Tamil
[te] – Telugu
[tr] – Turkish

# Attachments

## A Data

The list and the table in the following two subsections were adapted from Zeman et al. (2014) and updated.

### A.1 List of Treebanks in HamleDT

HamleDT currently covers the following 30 treebanks (data sizes are summarized in Table A.1):

- Arabic (ar): Prague Arabic Dependency Treebank[1] (PADT) 1.5 r349 / 2013 (Smrž et al., 2008)
- Basque (eu): Basque Dependency Treebank (BDT) (Aduriz et al., 2003)
- Bengali (bn): Hyderabad Dependency Treebank / ICON 2010 (Husain et al., 2010)
- Bulgarian (bg): BulTreeBank[2] / CoNLL 2006 (Simov and Osenova, 2005)
- Catalan (ca): AnCora-CA[3] / CoNLL 2009 (Taulé et al., 2008)
- Czech (cs): Prague Dependency Treebank 3.0[4] (Bejček et al., 2013)
- Danish (da): Danish Dependency Treebank[5] (DDT) (Kromann et al., 2004)
- Dutch (nl): Alpino Treebank[6] / CoNLL 2006 (Beek et al., 2002)
- English (en): Penn TreeBank[7] converted to dependencies / CoNLL 2009 (Surdeanu et al., 2008)
- Estonian (et): Eesti keele puudepank[8] (Bick et al., 2004)
- Finnish (fi): Turku Dependency Treebank[9] / 2011 (Haverinen et al., 2010)
- German (de): Tiger Treebank[10] / CoNLL 2009 (Brants et al., 2004)
- Greek (el): Greek Dependency Treebank (GDT) / CoNLL 2007 (Prokopidis et al., 2005)
- Greek, Ancient (grc): Ancient Greek Dependency Treebank[11] (AGDT) from the Perseus Project (Bamman and Crane, 2011)
- Hindi (hi): Hyderabad Dependency Treebank / COLING 2012 Shared Task (Husain et al., 2010)
- Hungarian (hu): Szeged Treebank[12] (SzTB) / CoNLL 2007 (Csendes et al., 2005)
- Italian (it): Italian Syntactic-Semantic Treebank[13] (ISST) / CoNLL 2007

[1] http://ufal.mff.cuni.cz/padt/
[2] http://www.bultreebank.org/indexBTB.html
[3] http://clic.ub.edu/corpus/
[4] http://ufal.mff.cuni.cz/pdt3.0/
[5] http://www.buch-kromann.dk/matthias/treebank/
[6] http://odur.let.rug.nl/~vannoord/trees/
[7] http://www.cis.upenn.edu/~treebank/
[8] http://www.cs.ut.ee/~kaili/Korpus/puud/
[9] http://bionlp.utu.fi/fintreebank.html
[10] http://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/tiger.html
[11] http://nlp.perseus.tufts.edu/syntax/treebank/
[12] http://www.inf.u-szeged.hu/projectdirs/hlt/index_en.html
[13] http://www.ilc.cnr.it/viewpage.php/sez=ricerca/id=874/vers=ing

(Montemagni et al., 2003)

- Japanese (ja): Tübingen Treebank of Spoken Japanese[14] (Tüba-J/S) / CoNLL 2006 (Kawata and Bartels, 2000)
- Latin (la): Latin Dependency Treebank[15] (LDT) from the Perseus Project (Bamman and Crane, 2011)
- Persian (fa): Persian Dependency Treebank[16] (PerDT) (Rasooli et al., 2011)
- Portuguese (pt): Bosque / Floresta sintá(c)tica[17] / CoNLL 2006 (Afonso et al., 2002)
- Romanian (ro): Resurse pentru Gramaticile de Dependenta[18] (Călăcean, 2008)
- Russian (ru): SynTagRus[19] / Russian Dependency Treebank (Boguslavsky et al., 2000)
- Slovak (sk): Slovak Treebank[20] (from the Slovak National Corpus) (Šimková and Garabík, 2006)
- Slovene (sl): Slovene Dependency Treebank[21] (SDT) / CoNLL 2006 (Džeroski et al., 2006)
- Spanish (es): AnCora-ES[22] / CoNLL 2009 (Taulé et al., 2008)
- Swedish (sv): Talbanken05[23] / CoNLL 2006 (Nilsson et al., 2005)
- Tamil (ta): Tamil Dependency Treebank v0.1[24] (TamilTB) (Ramasamy and Žabokrtský, 2012)
- Telugu (te): Hyderabad Dependency Treebank / ICON 2010 (Husain et al., 2010)
- Turkish (tr): METU-Sabanci (ODTÜ-Sabancı) Treebank[25] / CoNLL 2007 (Atalay et al., 2003)

[14] http://www.sfs.uni-tuebingen.de/en/tuebajs.shtml
[15] http://nlp.perseus.tufts.edu/syntax/treebank/
[16] http://dadegan.ir/en/perdt
[17] http://www.linguateca.pt/Floresta/principal.html
[18] http://www.phobos.ro/roric/texts/indexro.html
[19] http://www.ruscorpora.ru/en/search-syntax.html
[20] http://korpus.sk/
[21] http://nl.ijs.si/sdt/
[22] http://clic.ub.edu/corpus/
[23] http://stp.lingfil.uu.se/~nivre/research/Talbanken05.html
[24] http://ufal.mff.cuni.cz/~ramasamy/tamiltb/0.1/
[25] http://ii.metu.edu.tr/corpus

## A.2 Sizes

| Language | Primary tree type | Used data source | Sentences | Tokens | Avg. sent. length |
|---|---|---|---|---|---|
| Arabic (ar) | dep | CoNLL 2007 | 3043 | 116 793 | 38.38 |
| Basque (eu) | dep | primary | 11 226 | 151 604 | 13.50 |
| Bengali (bn) | dep | ICON 2010 | 1129 | 7252 | 6.42 |
| Bulgarian (bg) | phr | CoNLL 2006 | 13 221 | 196 151 | 14.84 |
| Catalan (ca) | phr | CoNLL 2009 | 14 924 | 443 317 | 29.70 |
| Czech (cs) | dep | primary | 87 913 | 1 503 738 | 17.04 |
| Danish (da) | dep | CoNLL 2006 | 5512 | 100 238 | 18.19 |
| Dutch (nl) | phr | CoNLL 2006 | 13 735 | 200 654 | 14.61 |
| English (en) | phr | CoNLL 2007 | 18 577 | 446 573 | 24.03 |
| Estonian (et) | phr | primary | 1315 | 9491 | 7.22 |
| Finnish (fi) | dep | primary | 4307 | 58 576 | 13.60 |
| German (de) | phr | CoNLL 2009 | 38 020 | 680 710 | 17.90 |
| Greek (el) | dep | CoNLL 2007 | 2902 | 70 223 | 24.20 |
| Greek (grc) | dep | primary | 21 160 | 308 882 | 14.60 |
| Hindi (hi) | dep | ICON 2010 | 3515 | 77 068 | 21.93 |
| Hungarian (hu) | phr | CoNLL 2007 | 6424 | 139 143 | 21.66 |
| Italian (it) | dep | CoNLL 2007 | 3359 | 76 295 | 22.71 |
| Japanese (ja) | dep | CoNLL 2006 | 17 753 | 157 172 | 8.85 |
| Latin (la) | dep | primary | 3473 | 53 143 | 15.30 |
| Persian (fa) | dep | primary | 12 455 | 189 572 | 15.22 |
| Portuguese (pt) | phr | CoNLL 2006 | 9359 | 212 545 | 22.71 |
| Romanian (ro) | dep | primary | 4042 | 36 150 | 8.94 |
| Russian (ru) | dep | primary | 34 895 | 497 465 | 14.26 |
| Slovene (sl) | dep | CoNLL 2006 | 1936 | 35 140 | 18.15 |
| Spanish (es) | phr | CoNLL 2009 | 15 984 | 477 810 | 29.89 |
| Swedish (sv) | phr | CoNLL 2006 | 11 431 | 197 123 | 17.24 |
| Tamil (ta) | dep | primary | 600 | 981 | 15.97 |
| Telugu (te) | dep | ICON 2010 | 1450 | 5722 | 3.95 |
| Turkish (tr) | dep | CoNLL 2007 | 5935 | 69 695 | 11.74 |

**Table A.1:** Data resources currently included in HamleDT.
The average sentence length is the number of tokens divided by the number of sentences. (In some treebanks, for example Bengali or Telugu, tokens represent sentence chunks; in some others, for example Arabic or Turkish, a single word can be represented by more than one token.)

# B Data Formats

## B.1 CoNLL

We borrow the description of the format and the table below from Buchholz and Marsi (2006): " All the sentences are in one text file and they are separated by a blank line after each sentence. A sentence consists of one or more tokens. Each token is represented on one line, consisting of 10 fields. Fields are separated from each other by a TAB. The 10 fields are: "

| Field number | Field name | Description |
|---|---|---|
| 1 | ID | Token counter, starting at 1 for each new sentence. |
| 2 | FORM | Word form or punctuation symbol. |
| 3 | LEMMA | Lemma or stem (depending on particular data set) of word form, or an underscore if not available. |
| 4 | CPOSTAG | Coarse-grained part-of-speech tag, where tagset depends on the language. |
| 5 | POSTAG | Fine-grained part-of-speech tag, where the tagset depends on the language, or identical to the coarse-grained part-of-speech tag if not available. |
| 6 | FEATS | Unordered set of syntactic and/or morphological features (depending on the particular language), separated by a vertical bar (\|), or an underscore if not available. |
| 7 | HEAD | Head of the current token, which is either a value of ID or zero ('0'). Note that depending on the original treebank annotation, there may be multiple tokens with an ID of zero. |
| 8 | DEPREL | Dependency relation to the HEAD. The set of dependency relations depends on the particular language. Note that depending on the original treebank annotation, the dependency relation may be meaningful or simply 'ROOT'. |
| 9 | PHEAD | Projective head of current token, which is either a value of ID or zero ('0'), or an underscore if not available. Note that depending on the original treebank annotation, there may be multiple tokens an with ID of zero. The dependency structure resulting from the PHEAD column is guaranteed to be projective (but is not available for all languages), whereas the structures resulting from the HEAD column will be non-projective for some sentences of some languages (but is always available). |
| 10 | PDEPREL | Dependency relation to the PHEAD, or an underscore if not available. The set of dependency relations depends on the particular language. Note that depending on the original treebank annotation, the dependency relation may be meaningful or simply 'ROOT'. |

## B.2 MST

The MST data format as described in `README` of the MSTParser (McDonald et al., 2005):

> Each sentence in the data is represented by 3 or 4 lines and sentences are space separated. The general format is:
>
> $w_1$ $w_2$ ... $w_n$
> $p_1$ $p_2$ ... $p_n$
> $l_1$ $l_2$ ... $l_n$
> $d_1$ $d_2$ ... $d_2$
> ...
>
> Where
>
> - $w_1$ ...$w_n$ are the n words of the sentence (tab delimited)
> - $p_1$ ...$p_n$ are the POS tags for each word
> - $l_1$ ...$l_n$ are the labels of the incoming edge to each word
> - $d_1$ ...$d_n$ are integers representing the position of each words parent

# C DVD

The enclosed DVD contains the files used in our experiments: the scripts we have written, a limited selection of some files produced in the experiments, and also third-party software – the DECCA scripts, the TreeTagger and the MSTParser. The contents of the DVD are structured to merge within a Treex installation, which is required for some of the scripts to work. For a detailed directory structure, see Figure C.1.

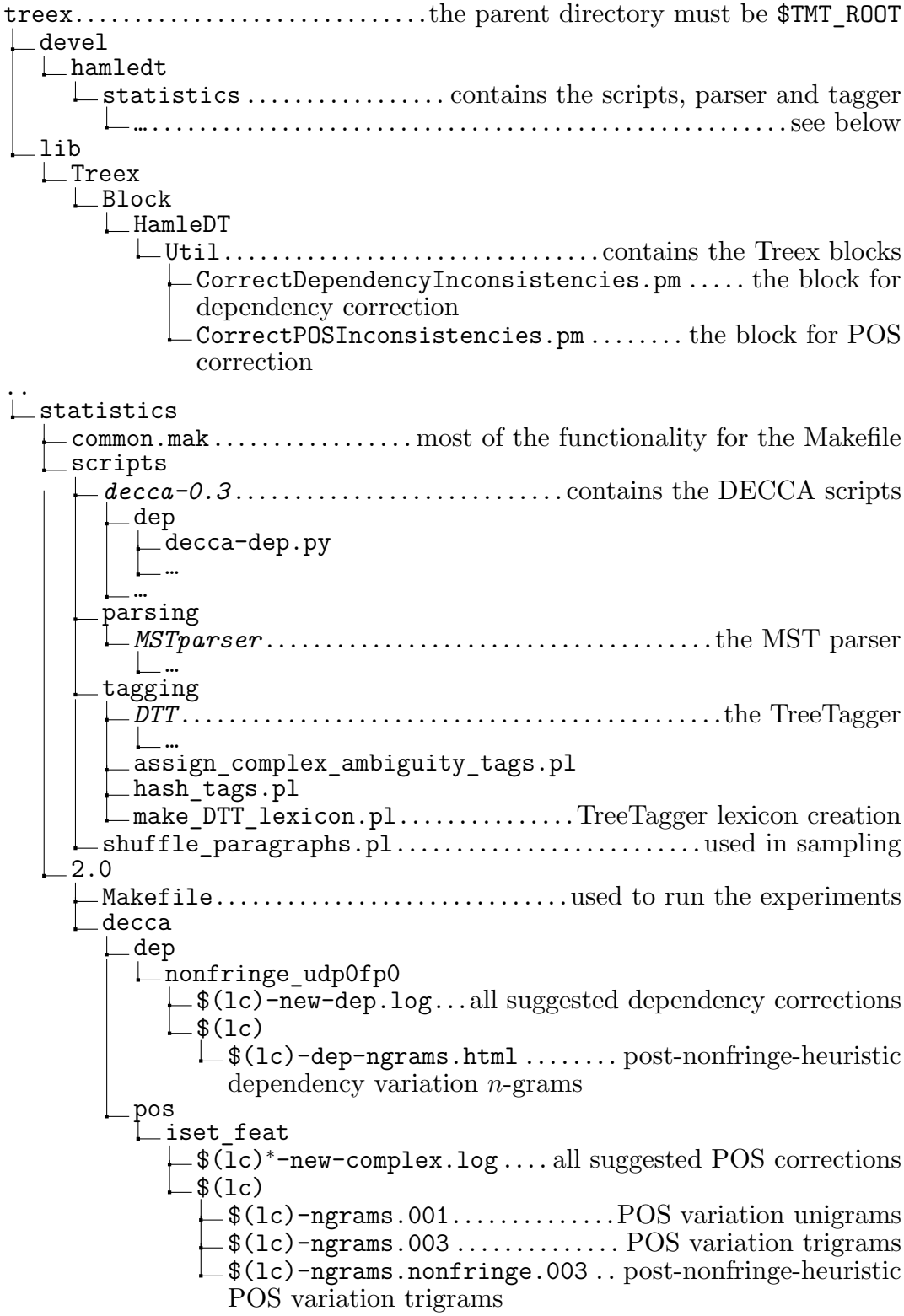



**Figure C.1:** DVD directory structure.
Top directories of third-party software are *emphasized*.

* language code