Bootstrapping UD treebanks for Delexicalized Parsing

Prasanth Kolachina
University of Gothenburg
prasanth.kolachina@gu.se

Aarne Ranta
University of Gothenburg
aarne@chalmers.se

Abstract

Standard approaches to treebanking traditionally employ a waterfall model (Sommerville, 2010), where annotation guidelines guide the annotation process and insights from the annotation process in turn lead to subsequent changes in the annotation guidelines. This process remains a very expensive step in creating linguistic resources for a target language, necessitates both linguistic expertise and manual effort to develop the annotations and is subject to inconsistencies in the annotation due to human errors.

In this paper, we propose an alternative approach to treebanking—one that requires writing grammars. This approach is motivated specifically in the context of Universal Dependencies, an effort to develop uniform and cross-lingually consistent treebanks across multiple languages. We show here that a bootstrapping approach to treebanking via interlingual grammars is plausible and useful in a process where grammar engineering and treebanking are jointly pursued when creating resources for the target language. We demonstrate the usefulness of synthetic treebanks in the task of delexicalized parsing, a task of interest when working with languages with no linguistic resources and corpora. Experiments with three languages reveal that simple models for treebank generation are *cheaper* than human annotated treebanks, especially in the lower ends of the learning curves for delexicalized parsing, which is relevant in particular in the context of low-resource languages.

1 Introduction

Treebanking remains a vital step in the process of creating linguistic resources for a language – a practice that was established in the last 2-3 decades (Marcus et al., 1994). The process of treebanking involves training human annotators in order to obtain high-quality annotations. This is a human-intensive and costly process where multiple iterations are performed to refine the quality of the linguistic resource. Grammar engineering is a complementary approach to creating linguistic resources: one that requires a different kind of expertise. These two approaches have remained orthogonal for obvious reasons: treebanks are primarily necessary to induce abstractions in NLU (Natural Language Understanding) models from data, while grammars are themselves abstractions arising from linguistic knowledge. Abstractions induced from data have proven themselves to be useful for robust NLU tasks, while grammars are better at precision tasks involving NLG (Natural Language Generation).

Given the resources required for treebanking, synthetic treebanks have been proposed and used as substitute in cross-lingual parsing for languages where treebanks do not exist. Such treebanks are created using parallel corpora where parse trees in one language are bootstrapped into a target language using alignment information through annotation projection (McDonald et al., 2011; Tiedemann, 2014) or using machine translation systems to bootstrap existing treebanks in one or more source language(s) to the target language (Tiedemann and Agic, 2016; Tyers et al., 2018). More recently, synthetic treebanks are generated for both real and artificial languages using multilingual treebanks by learning feasible parameter combinations (Wang and Eisner, 2016) – Wang and Eisner (2018) show that such treebanks can be useful to select the most similar language to train a parsing model for an unknown language.

At the same time, grammar-based treebanking approaches have been shown to work in monolingual setups—to derive rich linguistic representations defined by explicit grammars (Oepen et al., 2004). These approaches are carried out by parsing raw corpora with a target grammar and using an additional human disambiguation phase. Alternatively, existing treebanks are matched against the target grammar further reducing the human effort in disambiguation: these approaches face a challenge of under-specification in the source treebanks (Angelov, 2011). In the current paper, we propose a hybrid of these two methods: we use abstract syntax grammars as core linguistic abstraction to generate synthetic treebanks for a grammar that can be translated to target representations with high precision.

The question of annotation costs and ways to minimize the dependence on such annotated corpora has been a recurring theme in the field for the last two decades (Ngai and Yarowsky, 2000; Garrette and Baldridge, 2013). This question has also been extensively addressed in the context of dependency treebanks. We revisit this question in context of Universal Dependencies and recent work on the interplay between interlingua grammars and multilingual dependency trees in this scheme (Kolachina and Ranta, 2016; Ranta and Kolachina, 2017; Ranta et al., 2017). The use of interlingua grammars to bootstrap dependency treebanks guarantees two types of consistencies: multilingual treebank consistency and intra-treebank consistency. We study the efficacy of these dependency treebanks using learning curves of a transition-based parser in a *delexicalized parsing* setup. The delexicalized parsing setup allows for generation of parallel UD treebanks in multiple languages with minimal prerequisites on language-specific knowledge.

Another rationale behind the the current work in the context of cross-lingual parsing is while synthetic treebanks offer a "cheap" alternative, the signal for the target language is limited by the quality of the MT system. On the other hand, interlingua grammars provide a high-quality signal about the target language. High quality using interlingual grammars refers to accurate generation of word-order and morphology – although lexical selection in translation is still a problem. There have not been previous attempts in cross-lingual parsing to our knowledge studying the effect of these.

This paper is structured as follows: Section 2 gives the relevant background on interlingua grammars and the algorithm used to generate UD trees given treebank derived from an interlingua grammar. Section 3 describes our algorithm to bootstrap treebanks for a given interlingua grammar and parallel UD treebanks from them along with an intrinsic evaluation of these bootstrapped UD treebanks. Section 4 shows the parsing setup we use and Section 5 details the results of the parsing experiments.

2 Grammatical Framework

Grammatical Framework (GF) is a multilingual grammar formalism using abstract syntax trees (ASTs) as primary descriptions (Ranta, 2011). Originating in compilers, AST is a tectogrammatical tree representation that can be shared between languages. A GF grammar consists of two parts – an abstract syntax shared between languages and concrete syntax that is defined for each language. The abstract syntax defines a set of categories and a set of functions, as shown in Figure 1. The functions defined in the abstract syntax specify the result of putting subparts of two categories together and the concrete syntax specifies how the subparts are combined i.e. word-order preferences and agreement constraints specific to the language.

A comprehensive implementation of a multilingual grammar in GF is the **Resource Grammar Library**, GF-RGL (Ranta, 2009), which currently has concrete syntaxes for over 40 languages, ranging from Indo-European through Finno-Ugric and Semitic to East Asian languages. [1] This implementation contains a full implementation of the morphology of the language, and a set of 284 syntactic constructors that correspond to the core syntax of the language. Also included is a small lexicon of 500 lexical concepts from a set of 19 categories, of which 10 correspond to different subcategorization frames of verbs, 2 classes of nouns and adjectives. These grammars are *reversible*- i.e. they can be used for parsing and simultaneous multilingual generation into multiple languages. The concrete syntaxes for all the languages define the rules for these syntactic constructors and

[1] The current status of GF-RGL can be seen in `http://www.grammaticalframework.org/lib/doc/synopsis.html` which also gives access to the source code.

```
abstract syntax                          dependency configuration

PredVP  : NP   -> VP  -> Cl              nsubj  head
ComplV2 : V2   -> NP  -> VP              head   dobj
AdvVP   : VP   -> Adv -> VP              head   advmod
DetCN   : Det  -> CN  -> NP              det    head
ModCN   : AP   -> CN  -> CN              amod   head
UseN    : N     -> CN                    head
UsePron : Pron  -> NP                    head
PositA  : A     -> AP                    Head

cat A                                    ADJ
cat Adv                                  ADV
cat Det                                  DET
cat N                                    NOUN
cat Pron                                 PRON
cat V2                                   VERB
```

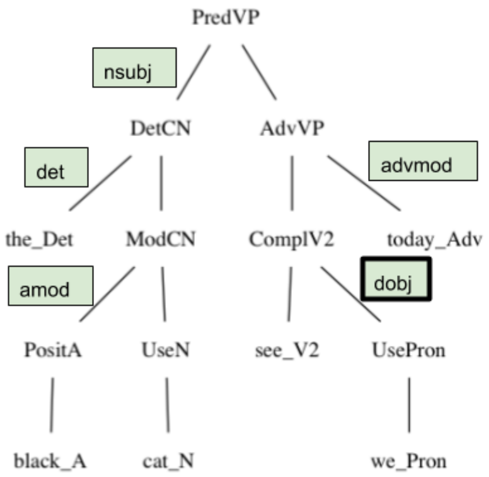


Figure 1: Abstract syntax of a GF grammar and its specification for UD scheme. Also shown is an example AST for the sentence *the black cat sees us today*. Any function with a definition written as $\mathrm{f} : \mathrm{C}_1 \rightarrow \mathrm{C}_2 \rightarrow ...\mathrm{C}_\mathrm{n} \rightarrow \mathrm{C}$; can be rewritten as a context-free rule f. $\mathrm{C} ::= \mathrm{C}_1\mathrm{C}_2...\mathrm{C}_\mathrm{n}$.

the lexical concepts. The expressivity of these grammars is equivalent to a PMCFG (Seki et al., 1991), which makes parsing complexity of this formalism polynomial in sentence length. Polynomial parsing with high exponents can still be too slow for many tasks, and is also brittle if the grammars are designed to not over-generate. But generation using GF grammars has been shown to be both precise and fast, which suggests the idea of combining data-driven parsing with grammar-driven generation. We refer the interested reader to Ljunglöf (2004) for discussion on expressivity of this formalism and Angelov (2011); Angelov and Ljunglöf (2014) for discussion on probabilistic parsing using GF grammars.

2.1 gf2ud

Kolachina and Ranta (2016) propose an algorithm to translate ASTs to dependency trees, that takes a specification of the abstract syntax of the GF grammar (referred to as *configurations*, see Figure 1) which describes the mapping between the grammar and a target dependency scheme, in this case Universal Dependencies. These configurations can be interpreted as a synchronous grammar over the abstract syntax as source and dependency scheme as target.

The first step in this transducer is a recursive annotation that marks for each function in the AST, one of the arguments as `head` and specifies labels for the other arguments, as specified by the configuration. The algorithm to extract the resulting dependency tree from the *annotated AST* is simple.

- for each leaf X (which corresponds to a lexical item) in the AST
  - trace the path up towards the root until you encounter a label L
  - from the node immediately above L, follow the **spine** (the unlabeled branches) down to another leaf Y
  - Y is the head of X with label L

At the end of these two steps, the resulting data-structure is an *abstract dependency tree* (ADT shown in Figure 2). It should be noted that the order of nodes shown in the ADT does not reflect the surface order that is specific to a language. The ADT combined with the concrete syntax of a language and concrete configurations (when necessary) results in the corresponding full UD tree. The concrete configurations are necessary to provide appropriate labels to syncategorematic words like auxiliary verbs and negation particles. Additionally, the category configuration on the abstract

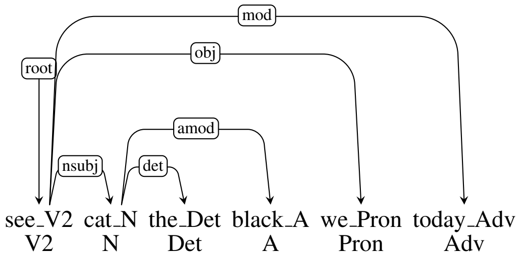


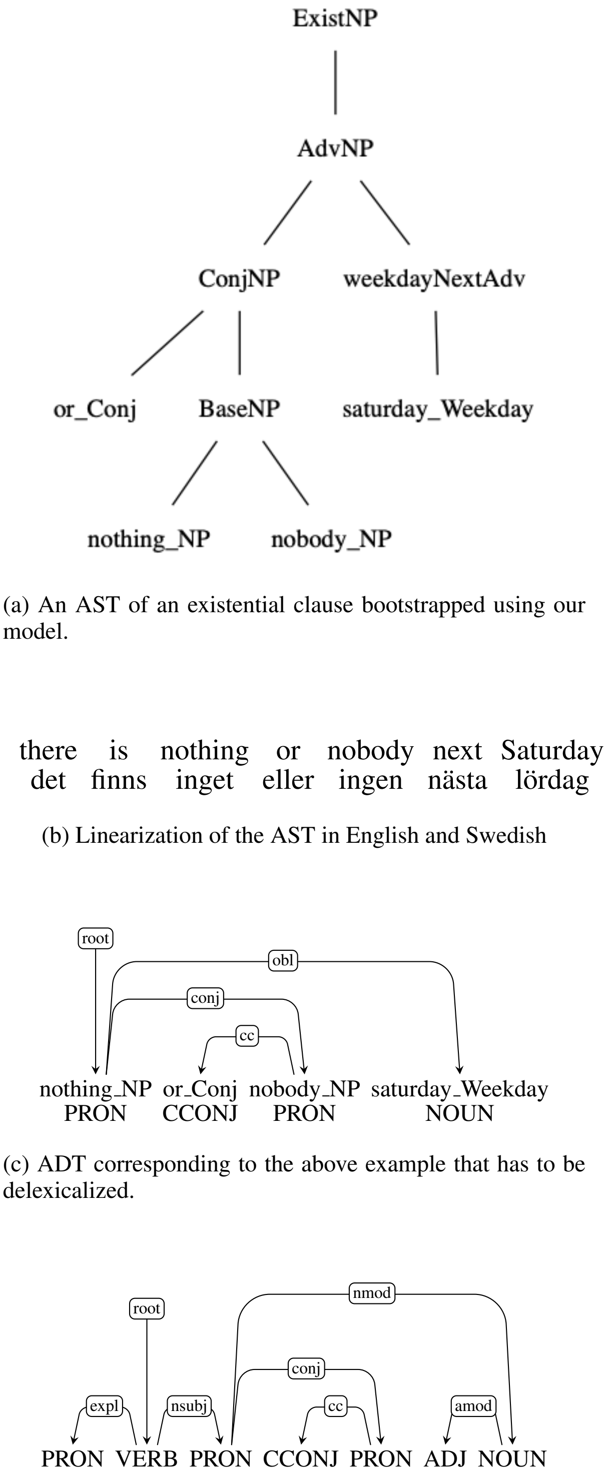
Figure 2: ADT for the sentence *the black cat sees us today*. The nodes in the ADT correspond to lexical functions defined in the grammar. Also shown is the UD part-of-speech tag sequence. Note that the order of nodes does not reflect the surface order in any particular language.

syntax can be augmented with a language-specific category configurations to generate the morphological features in the dependency tree with a desired tag set.

Kolachina and Ranta (2016) show that their method can be used to generate partially labeled UD trees for 30 languages when the corresponding concrete syntax is available. They also show that using configurations defined on abstract syntax alone and depending on the availability of the concrete syntax, a large fraction (around 75–85% of edges) of the dependency treebanks can be generated automatically. This is done with small treebanks of ASTs – a UD treebank of 104 ASTs and a GF treebank of 400 ASTs. Their results show that parallel UD treebanks can be bootstrapped using ASTs and interlingua grammars, the usefulness of such treebanks however is not addressed in that work. Full UD treebanks can be generated when concrete configurations (those addressing syncategorematic words) are additionally available for the language.

3 Bootstrapping AST and UD treebanks

The abstract syntax component of a GF grammar is an algebraic datatype definition, which can also be seen as a context-free grammar (CFG). The disambiguation model defined in GF uses a context-free probability distribution defined on the abstract syntax. The advantage of defining the distribution on the abstract syntax is that it allows for transfer of distribution to languages for which GF treebanks do not exist. The context-free distribution decomposes the probability estimate of a tree as the product of probabilities of the sub-trees and the probability of the function applied to these sub-



(a) An AST of an existential clause bootstrapped using our model.

(b) Linearization of the AST in English and Swedish

(c) ADT corresponding to the above example that has to be delexicalized.

(d) The delexicalized UD tree in both English and Swedish shares the same part-of-speech tag sequence and dependency labels

Figure 3: Example of a bootstrapped AST and UD tree and the intermediate ADT.

trees. The probabilistic abstract syntax grammar can therefore be defined in terms analogous to a probabilistic CFG (PCFG). The probability distribution over the set of categories in the grammar is also included in the distribution corresponding to the abstract syntax.

We use this formulation as a starting point and generate ASTs for a given grammar. The ASTs bootstrapped using the probability model defined above are correct in terms of the grammar but do not follow the selectional preferences that encode semantic preferences verbs have for their arguments typically found in language. For this reason, we refer to the bootstrapped treebanks as "synthetic" data.

Additionally, while the algorithm used to bootstrap ASTs does not change depending on whether the grammar includes a lexicon or not, it is significantly faster depending the size of the grammar. Stacking `gf2ud` defined using abstract configurations on top of these bootstrapped ASTs results in a treebank of ADTs. Alternatively, the concrete syntax of a language can straightforwardly be used to *linearize* a corpus of the target language. The concrete syntax and the concrete configurations when available are used to generate fully labelled UD treebanks for a target language. Figure 3 shows an example of a synthetic AST and delexicalized UD tree bootstrapped using the RGL.

The bootstrapping algorithm uses a parameter corresponding to the maximum depth of the trees d to be generated. The generative story is as follows:

- Pick a category C using the distribution over categories defined in the probability model.
- Select a function F with the definition $\mathrm{C}_1 \rightarrow \mathrm{C}_2 \rightarrow ... \rightarrow \mathrm{C}_\mathrm{n} \rightarrow \mathrm{C}$ according to the conditional distribution $P(F|C)$.
- Recursively apply the same step to build subtrees of maximum depth $d-1$, $\mathrm{t}_{\mathrm{C}_1}$, $\mathrm{t}_{\mathrm{C}_2}$ ... $\mathrm{t}_{\mathrm{C}_\mathrm{n}}$ of categories C_1, C_2 ... C_n respectively.
- Return (F $\mathrm{t}_{\mathrm{C}_1}$ $\mathrm{t}_{\mathrm{C}_2}$ .. $\mathrm{t}_{\mathrm{C}_\mathrm{n}}$).

3.1 Differences against UDv2

The design of the RGL and corresponding configurations do not contain all of the structures defined in the UD annotation scheme. The missing structures fall into two major categories: labels that depend on the lexical realization in a specific language, and structures that correspond to specific linguistic constructions that are not part of the core RGL syntax. Examples of the first type include multi-word expressions and proper nouns (labeled using `fixed` and `flat` label). In the second class, are ellipsis and paratactic constructions in addition to labels that are used in robust analysis of web text (`orphan`, `goeswith` and `reparandum`). Examples that cover these labels can be generated by re-writing the grammar: however, we found very few instances of these in the treebanks. Finally, another variation in the bootstrapped treebanks is in the case of label subtypes that are optionally defined in a language-specific manner. While the configurations allow for accurate generation of certain labels (e.g. `obl:agent` in the case of passive agents), recovering similar information in other instances is not possible without a significant redesign of the RGL (e.g. `obl:tmod` for temporal modifiers). We address this issue by restricting `gf2ud` to generate only the core labels in UD and ignore subtype labels uniformly across languages.

Language	H(P_{UD})	H(P_{GF})	Cross-entropy
Afrikaans	39.59	58.34	63.12
Arabic	40.00	42.13	51.38
Basque	44.19	51.19	54.21
Bulgarian	32.09	53.76	61.23
Catalan	44.49	49.37	57.39
Chinese	39.25	42.10	59.76
Danish	44.85	55.28	63.39
Dutch	48.99	49.67	61.27
English	50.52	45.31	58.17
Estonian	39.45	43.82	49.35
Finnish	47.86	41.52	54.39
French	43.41	49.43	53.47
German	41.35	49.35	51.29
Greek	29.48	41.13	49.17
Hindi	32.99	43.18	54.27
Italian	38.55	51.37	59.64
Japanese	27.34	40.18	47.25
Latin	42.07	43.47	49.89
Latvian	49.75	49.91	59.26
Norwegian (bokmal)	40.29	45.97	53.17
Norwegian (nynorsk)	37.29	44.56	56.32
Persian	33.07	47.29	47.16
Polish	23.85	41.27	49.83
Portuguese	40.84	48.73	53.60
Romanian	47.31	52.31	57.12
Russian	39.14	47.92	52.84
Spanish	46.36	52.17	57.73
Swedish	35.36	47.41	51.39
Urdu	33.70	42.14	58.73
Icelandic	N/A	51.26	N/A
Thai	N/A	41.23	N/A

Table 1: Entropy values of probability distributions P(label—(head-pos)) for different languages estimated from real (P_{UD}) and bootstrapped (P_{GF}) treebanks. If a language has more than one treebank in the UD distribution, we select one treebank as the primary treebank and use that to estimate the distribution and in the parsing experiments. Languages for which a UD treebank does not exist but is included in GF-RGL are listed towards the bottom of the table.

Table 1 shows the entropies of the conditional probability distribution defined as probability of a UD label given the part-of-speech tag of the head.

The distributions are estimated on both the synthetic UD treebank and a human annotated UD treebank [2] Also shown in the table are the cross-entropy values between the distribution estimated from the synthetic and the original treebanks.

4 UD Parsing

The bootstrapped UD treebanks are used to train delexicalized parsing models. We choose to work with the delexicalized UD treebanks for two reasons: first, the context-free assumption in the probabilistic model defined on the abstract syntax makes the tree generation decomposable, but selectional preferences are not encoded in the generative model used for bootstrapping the ASTs. Secondly, generating a full UD treebank assumes the availability of an interlingua lexicon – which reduces the portability of this approach to new languages.[3] For both these reasons, we restrict ourselves to strictly delexicalized UD treebanks in our parsing experiments.

We are interested in the following three use-cases depending on the size of the training data (N) available for inducing parsing models.

- When N $\leq$ 1K sentences[4] are available for a language. There are around 20 treebanks in the current UD distribution that match this criterion and almost all these treebanks have been manually annotated from scratch. This corresponds to the scenario of under-resourced languages, where either the monolingual corpus for treebank or annotators for treebanking are scarce. This scenario strongly corresponds to our proposed idea of simultaneous grammar engineering and treebanking.

- When 1K $\leq$ N $\leq$ 5K sentences[5] are available for a language. There are around 18 treebanks in the current UD distribution that match this criterion. While one can argue that these languages are not really under-resourced, this setup matches the typical case of training domain-specific parsers either for a particular domain like bio-medical or legal texts.

- The case where treebanks are larger than either of the two previous scenarios N $\geq$ 5K. This setup is interesting to test the limit of how useful are bootstrapped ASTs and UD treebanks to train parsing models.

For each of these use-cases, we train parsing models using data from both human annotated UD treebanks and synthetic treebanks for different sizes of training data. The resulting parsing models are evaluated using labelled attachment scores, obtained by parsing the test set of the UD treebank for the language in question. We experiment with an off-the-shelf transition-based dependency parser that gives good results in the dependency parsing task (Straka and Straková, 2017). In the ideal case the experiments need to be carried out using multiple parsers from both the transition-based and graph-based paradigms. We leave that for future work.

5 Experiments

We ran experiments with 3 languages – English, Swedish and Finnish in this paper. In addition to the availability of a concrete syntax for the language, our approach also requires concrete configurations for the languages (Kolachina and Ranta, 2016) in order to bootstrap full UD trees. Table 2 shows statistics about the concrete configurations for the RGL grammar for the languages. The probability distribution defined on the RGL was estimated using the GF-Penn treebank (Marcus et al., 1994; Angelov, 2011) of English. This raises another question – how well does the distribution defined on the abstract syntax of the RGL estimated from monolingual data transfer across other languages. The bootstrapping algorithm was restricted to generate 20K ASTs of depth less than 10.[6]

We use UDPipe (Straka and Straková, 2017) to train parsing models, using comparable settings to the baseline systems provided in the CoNLL18 shared task (Zeman and Hajič, 2018). Gold tokenization and part-of-speech tags are used in both training and testing the parser. This was done to control for differences in tagging performance across the synthetic and original UD treebanks. The models are trained using the primary treebanks from Universal Dependencies v2.3 distribution.[7] We plot the learning curves for parsing

[2] The UD treebanks are taken from the v2.3 distribution.

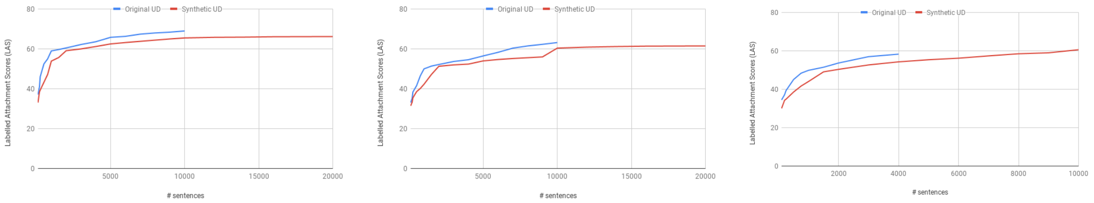
[3] There is ongoing work on developing interlingual lexica from linked data like WordNet (Virk et al., 2014; Angelov and Lobanov, 2016).

[4] This approximately corresponds to 20K tokens.

[5] This approximately corresponds to 20K – 100K tokens.

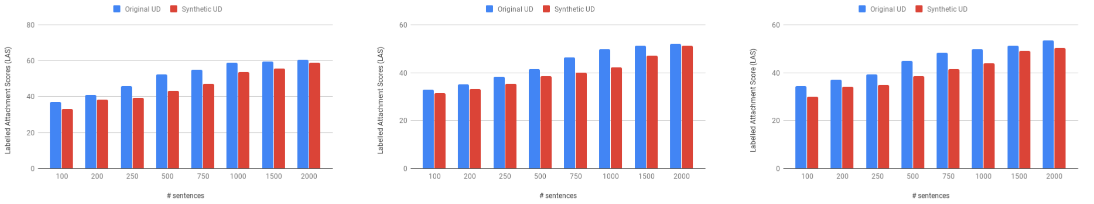
[6] Trees of depth less than 4 were filtered out in the process.

[7] The notion of primary treebank for a language has been



(a) Learning curves for English (b) Learning curves for Finnish (c) Learning curves for Swedish

Figure 4: Learning curves for parsing models of trained on original UD and synthetic UD treebanks.



(a) Learning curves for English (b) Learning curves for Finnish (c) Learning curves for Swedish

Figure 5: Learning curves shown using bar plots for parsing models trained on less than 1000 sentences from original UD and 2000 sentences from synthetic UD treebanks.

Language	Abstract	Concrete	Morph-features
English	143	21	57
Swedish	143	25	59
Finnish	143	31	57

Table 2: Estimate of the effort required in `gf2ud`. The abstract configurations are the same for all languages, while the concrete functions and morph-features are defined for each language. The first column corresponds to configurations for syntactic constructors in the RGL, and second column corresponds to constructors that use syncategorematic words in the linearization.

models in Figure 4 trained on both the original and synthetic treebank data for each use case outlined in Section 4. The learning curves were plotted using the LAS accuracies obtained on the test set for the three languages using models trained on both the original and the synthetic treebanks. It is seen from the learning curves that models trained on the synthetic treebanks do not outperform the models trained using original UD treebanks.

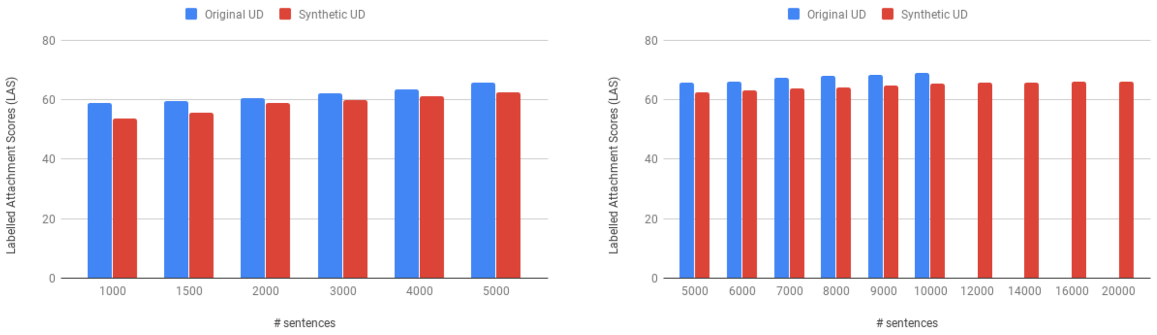
However, the full learning curves shown in Figure 4 do not tell the complete story. Figure 5 shows the learning curves (visualized using bar plots) for English, Finnish and Swedish in the setup where less than 1K sentences from UD treebanks are used. It is clear from the plots for all the three languages that the synthetic treebanks are sub-optimal when directly compared against real treebanks of the **same size**. However, what is interesting is that parsing models in this range (i.e. N $\leq$ 1K) with synthetic treebanks quickly reach comparable accuracies to using real treebank data, with an approximate effective data coefficient of 2.0. In other words comparable accuracies can be obtained using roughly twice the amount of synthetic data, generated for free by the abstract syntax grammar.

It is interesting to note that the learning curves using the synthetic data for the English parsing models become comparably flat in our setup with less than 5K sentences (shown in Figure 6a). Despite the lower improvements with increasing treebank sizes, there is still a consistent improvement in parsing accuracies with the best accuracy of 65.4 LAS using 10K synthetic samples (shown in Figure 6b). This pattern is consistent across Swedish and Finnish, which allows us to draw the conclusion that while the effective data coefficient is smaller, the synthetic treebanks are still useful to improve parsing accuracies.

6 Related Work

The current trend in dependency parsing is directed towards using synthetic treebanks in an attempt to cover unknown languages for which

made obsolete in UD v2.3 distribution - with all treebanks being assigned a code. So, we use the term primary in this paper to refer to EWT for English, TDT for Finnish and Talbanken for Swedish.



(a) Learning curves for English with N between 1K and 5K samples

(b) Learning curves for English with N between 5K and 10K samples

Figure 6: Learning curves shown using bar plots for parsing models of English

resources are minimal or do not exist altogether. Such treebanks rely on various auxiliary resources: parallel corpora (Tiedemann, 2014), multilingual word-embeddings (Xiao and Guo, 2014), MT system for the target language (Tiedemann and Agic, 2016; Tyers et al., 2018) or more minimally, tagged corpora in the target language (Wang and Eisner, 2018).

Tiedemann and Agic (2016) propose a method to generate synthetic treebanks for new languages using machine translation systems to transfer cross-linguistic information from resource-rich language to under-resourced languages. This work builds on top of many previous approaches to cross-lingual parsing using parallel corpora and multilingual word-embeddings. The synthetic treebanks generated in the current work are are different in two ways:

- we assume multilingual abstraction and the concrete syntaxes are available, namely the GF-RGL to generate language-independent samples in the form of ASTs.
- we also assume that a distribution of the target language is not available and what is available is a distribution on the abstract syntax that generalizes to other languages.

Hence, the resulting treebank is licensed by a grammar, and high-precision cross-linguistic information is specified, but the distribution over the resulting treebank is different from the distribution obtained using the real treebanks. An alternative to the method of bootstrapping UD treebanks is to use `ud2gf` (Ranta and Kolachina, 2017) as a way to translate existing UD treebanks to GF treebanks, that are licensed by a grammar.

The current work also relates to more recent work in data-augmentation for dependency parsing (Sahin and Steedman, 2018) and more generally in NLP (Sennrich et al., 2016). The augmentation methods are designed to address data scarcity by exploiting monolingual corpora or generating synthetic samples in multilingual applications. However, the underlying abstractions used to generate the synthetic data are induced from auxiliary corpora.

Jonson (2006) show that synthetic corpora generated using a GF grammar can be used to build language models for speech recognition. Experiments in their work show that synthetic in-domain examples generated using the grammar when combined with large out-of-domain data result in significant reduction of word error rate of the speech recognizer. This work falls in line with similar approaches to combine corpus driven approaches with rule-based systems (Bangalore and Johnston, 2004), as a way to combine the statistical information available from corpora with good coverage resulting from rule-based abstractions especially when working with restricted domains. In this paper, we restrict ourselves to utilizing synthetic treebanks for parsing, and leave the discussion on ways to combine synthetic treebanks with real treebanks as future work. This choice is primarily motivated by our interest in grammar-based development of dependency treebanks as opposed to the traditional way of treebanking – by training human annotators.

7 Conclusions

In the current paper, we propose an alternative approach to cross-lingual treebanking — one that recommends grammar engineering. Multilingual abstractions that facilitate bootstrapping of cross-lingual treebanks have been previously explored in the setup of low precision high recall methods. These methods presume the availability of different resources in order to induce the cross-linguistic signal – parallel or multilingual corpora, word embeddings etc. Our approach explores the opposite direction – multilingual grammars of high precision are used to bootstrap parallel treebanks. While these multilingual grammars are not easy to develop, the question of how useful such grammars are is one that has been largely unexplored in the context of cross-lingual syntactic parsing.

We use a context-free probability model to generate ASTs that are used to bootstrap parallel UD treebanks in 3 languages. Experiments in delexicalized parsing show that these treebanks are useful in two scenarios – when data in the target language is minimal (<1K sentences) and small (<5K sentences). In the future, we intend to look at ways to generate synthetic treebanks from existing UD treebanks of languages using `ud2gf` (Ranta and Kolachina, 2017), that aims to address the lack of syntactic distributions in our synthetic treebanks. We also did not pursue the obvious direction of combining the real and synthetic treebanks in the current work: we leave this for future work. Another direction that is of interest is to augment existing treebanks with syntactic variations to quantify the need for regular syntactic variants in parser development, such as converting declaratives to questions, varying tense and polarity, adding and removing modifiers, and so on. String-based augmentation (as opposed to precise grammar-based generation) in this direction has already shown promising results (Sahin and Steedman, 2018).

Acknowledgements

We want to thank Joakim Nivre, Richard Johansson, Filip Ginter, Lilja Øvrelid and Marco Kuhlmann for the discussion and the anonymous reviewers for helpful comments on this work. The project has been funded by the REMU project (Reliable Multilingual Digital Communication, Swedish Research Council 2012-5746).

References

Krasimir Angelov. 2011. *The Mechanics of the Grammatical Framework*. Ph.D. thesis, Chalmers University of Technology.

Krasimir Angelov and Peter Ljunglöf. 2014. Fast Statistical Parsing with Parallel Multiple Context-Free Grammars. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 368–376, Gothenburg, Sweden. Association for Computational Linguistics.

Krasimir Angelov and Gleb Lobanov. 2016. Predicting Translation Equivalents in Linked WordNets. In *Proceedings of the Sixth Workshop on Hybrid Approaches to Translation (HyTra6)*, pages 26–32, Osaka, Japan. The COLING 2016 Organizing Committee.

Srinivas Bangalore and Michael Johnston. 2004. Balancing data-driven and rule-based approaches in the context of a Multimodal Conversational System. In *HLT-NAACL 2004: Main Proceedings*, pages 33–40, Boston, Massachusetts, USA. Association for Computational Linguistics.

Dan Garrette and Jason Baldridge. 2013. Learning a Part-of-Speech Tagger from Two Hours of Annotation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 138–147, Atlanta, Georgia. Association for Computational Linguistics.

Rebecca Jonson. 2006. Generating Statistical Language Models from Interpretation Grammars in Dialogue Systems. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

Prasanth Kolachina and Aarne Ranta. 2016. From Abstract Syntax to Universal Dependencies. *Linguistic Issues in Language Technology*, 13(3).

Peter Ljunglöf. 2004. *The Expressivity and Complexity of Grammatical Framework*. Ph.D. thesis, Department of Computing Science, Chalmers University of Technology and University of Gothenburg.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyr, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating Predicate Argument Structure. In *HUMAN LANGUAGE TECHNOLOGY: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, pages 114–119.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-Source Transfer of Delexicalized Dependency Parsers. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 62–72, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Grace Ngai and David Yarowsky. 2000. Rule Writing or Annotation: Cost-efficient Resource Usage for Base Noun Phrase Chunking. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 117–125, Hong Kong. Association for Computational Linguistics.

Stephan Oepen, Dan Flickinger, Kristina Toutanova, and Christopher D. Manning. 2004. LinGO Redwoods: A Rich and Dynamic Treebank for HPSG. *Research on Language and Computation*, 2(4):575–596.

Aarne Ranta. 2009. The GF Resource Grammar Library. *Linguistic Issues in Language Technology*, 2(2).

Aarne Ranta. 2011. *Grammatical Framework: Programming with Multilingual Grammars*. CSLI Publications, Stanford.

Aarne Ranta and Prasanth Kolachina. 2017. From Universal Dependencies to Abstract Syntax. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 107–116, Gothenburg, Sweden. Association for Computational Linguistics.

Aarne Ranta, Prasanth Kolachina, and Thomas Hallgren. 2017. Cross-Lingual Syntax: Relating Grammatical Framework with Universal Dependencies. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 322–325, Gothenburg, Sweden. Association for Computational Linguistics.

Gozde Gul Sahin and Mark Steedman. 2018. Data Augmentation via Dependency Tree Morphing for Low-Resource Languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5004–5009, Brussels, Belgium. Association for Computational Linguistics.

Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191–229.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Ian Sommerville. 2010. *Software Engineering*, 9th edition. Addison-Wesley Publishing Company, USA.

Milan Straka and Jana Straková. 2017. Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Jörg Tiedemann. 2014. Rediscovering Annotation Projection for Cross-Lingual Parser Induction. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1854–1864, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Jörg Tiedemann and Zeljko Agic. 2016. Synthetic Treebanking for Cross-Lingual Dependency Parsing. *The Journal of Artificial Intelligence Research (JAIR)*, 55:209–248.

Francis Tyers, Mariya Sheyanova, Aleksandra Martynova, Pavel Stepachev, and Konstantin Vinogorodskiy. 2018. Multi-source synthetic treebank creation for improved cross-lingual dependency parsing. In *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 144–150, Brussels, Belgium. Association for Computational Linguistics.

Shafqat Mumtaz Virk, K.V.S Prasad, Aarne Ranta, and Krasimir Angelov. 2014. Developing an interlingual translation lexicon using WordNets and Grammatical Framework. In *Proceedings of the Fifth Workshop on South and Southeast Asian Natural Language Processing*, pages 55–64, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Dingquan Wang and Jason Eisner. 2016. The Galactic Dependencies Treebanks: Getting More Data by Synthesizing New Languages. *Transactions of the Association for Computational Linguistics*, 4:491–505.

Dingquan Wang and Jason Eisner. 2018. Synthetic Data Made to Order: The Case of Parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1325–1337, Brussels, Belgium. Association for Computational Linguistics.

Min Xiao and Yuhong Guo. 2014. Distributed Word Representation Learning for Cross-Lingual Dependency Parsing. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 119–129, Ann Arbor, Michigan. Association for Computational Linguistics.

Daniel Zeman and Jan Hajič. 2018. Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. Brussels, Belgium. Association for Computational Linguistics.