Master thesis project planning report

Efficient conversion from Dependency Trees to Abstract Syntax Trees in Natural Language Processing

Andreas Källberg anka.213@gmail.com

Supervisor at CSE: Aarne Ranta

Relevant completed courses student 1:

TIN092 Algorithms
TDA342 Advanced functional programming
DAT151 Programming language technology

March 15, 2023

1 Introduction

Something general about rule based NLG and NLP and computer grammars

1.1 Dependency trees and Universal Dependencies

A dependency tree is a tree structure that shows the grammatical relationship between words in a sentence. Each word becomes a node in the tree with the main verb as the root and the edges represents the relation and the direction of the dependency.

For example in the sentence ”John ate an apple”, the word ”ate” would become the root and ”John” and ”apple” would be direct children, where the arrow from ”ate” to apple would be labeled as the subject relation while the arrow to ”apple” would be labeled as the object. There would additionally be an arrow from ”apple” to ”an” which marks that ”an” is a determiner for ”apple”.

Add an image here

The specific standard for dependency trees that this paper is about is called Universal Dependencies (UD). UD is based on the idea of making a multilingual standard for dependency trees where the same set of tags can be used regardless of which language the sentence is written in.

Talk about the machine learning aspect of UD

1.2 Abstract trees and Grammatical Framework

Another way of describing the grammatical structure of a sentence is through abstract syntax trees, where instead of using words as the nodes in the tree, you use

Some basic intro on abstract trees

Write about abstract trees

consider swapping order

Grammatical Framework[3] (GF) is a formalism for describing natural language grammars as code, which allows converting between natural language and a language-independent abstract syntax. It is split into a generic resource grammar that covers morphology and the language as a whole and application grammars for a more narrow domain which allows a more semantic abstract syntax. When you try to write a grammar that covers a very wide domain, you will often get an over-generating grammar where each sentence can be parsed in very many ways into many different trees, where usually only one is the intended way.

1.3 Strengths and weaknesses

While the machine-learning based approach of UD allows it to guess the correct tree better for ambiguous sentences and allows it to handle grammatically incorrect sentences better, GF is much more capable when it comes to performing transformations on the sentences, while maintaining correct morphology and grammar. This makes it attractive to parse sentences using UD and then convert the parsed trees to GF trees in order to perform further transformations.

1.4 Applications

gf-ud can be useful for both translation and semantics[4]. Another application has been in concept alignment[2]. There has also been work using it for analysis of law text for the purpose of making a Controlled Natural Language for law[1].

actually explain this

2 Background and Problem

3 Goals and Challenges

4 Limitations

Only the direction of ud2gf is studied here.

why?

other limitations?

5 Approach

5.1 Performance

1. Finding the main source of slowness, which was done with profiling.

2. Analysing the current algorithms, which are based on brute-force, trying all combinations, with some simple filtering.

3. Finding a better algorithm, which avoids exploring paths that could never be the correct answer and which avoids duplicate work.

4. Analysing algorithmic complexity of both algorithms and testing the practical performance to confirm the results.

5.2 Flexibility

In order to allow changing the shape of trees when translating from UD to GF, the macro language needs to be expanded. A first prototype of this can be done with minimal code changes by making macro expansion recursive and then representing the code for the transformation in Church-encoding, inspired by lambda-calculus.

This approach can be evaluated by seeing how well it covers different tree shape changes for different trees one would encounter.

It could also be worthwhile to make a more user-friendly version of the advanced macros that can be understood without knowing about Church-encoding

5.3 Debugging tool

Go through each component of the algorithms in order to find where applying a rule can go wrong and add detection for them. Additionally try out the debugging tool on a real grammar, e.g. in the context of [1], in order to find edge-cases which were not handled by the debugging tool.

5.4 Linearization-aware translation

Here we need a way to determine which linearization is actually closer and when to keep multiple options for a later stage of the translation. Some care also needs to be taken in determining which trees can be tossed away and which ones need to be kept around for a later stage.

Evaluating if the results are better with this version can be done in a large part by comparing the input string with the resulting string after translating to GF and linearizing.

6 Evaluation

One method for evaluation is through synthetic experiments of generating random GF trees through GF's built in functionality, then translating those to UD trees and then back to GF again. There are several different aspects that could be evaluated here:

- Completeness: the ability to always get complete trees and no Backups
- Accuracy: How well the tree matches the original after a roundtrip
- Perfomance: How fast the code runs
- Error analysis:

What does this mean?

7 Risk analysis and ethical considerations

There are very little risks to speak of as the work is

There are no ethical considerations to speak of, since ...

(possibly evaluation data for the project and training data for the UD models)

8 Time plan

Most of the practical implementation work has already been completed by the authors of this thesis prior to the official start of this thesis. The main parts that remain is evaluation and writing the report itself. Optionally, if there is time, some additional work to further improve the results can be performed (which ones?).

The plan is to complete the thesis before the summer of 2023.

Todo: more detailed plan

9 References

References

[1] Inari Listenmaa, Maryam Hanafiah, Regina Cheong, and Andreas Källberg. Towards CNL-based verbalization of computational contracts. In *Proceedings of the Seventh International Workshop on Controlled Natural Language (CNL 2020/21)*, Amsterdam, Netherlands, September 2021. Special Interest Group on Controlled Natural Language.

[2] Arianna Masciolini and Aarne Ranta. Grammar-based concept alignment for domain-specific machine translation. In *Proceedings of the Seventh International Workshop on Controlled Natural Language (CNL 2020/21)*, 2021.

[3] A. Ranta. Grammatical Framework: A Type-Theoretical Grammar Formalism. *The Journal of Functional Programming*, 14(2):145–189, 2004.

[4] Aarne Ranta, Krasimir Angelov, Normunds Gruzitis, and Prasanth Kolachina. Abstract syntax as interlingua: Scaling up the grammatical framework from controlled languages to robust pipelines. *Computational Linguistics*, 46(2):425–486, 2020.