TDDE09 - Project Plan

Joar Måhlén, Martin Brolin, Ahsan Iqbal, Anass Benkirane

February 2021

1 Background

This project will investigate syntactic parsing, the task of mapping a sentence to a formal representation of its syntactic structure. The structure is represented in the form of dependency trees meaning that words are connected to each other through asymmetric relations were one word is a head and another is a dependent. The relation/arcs may also be labelled to describe the type of the syntactic relation between the words. Dependency parsing can be divided in two different paradigms, graph-based dependency parsing and transition-based dependency parsing. In transition-based dependency parsing you cast parsing as a sequence of local classification problems where you predict a certain parser action/move based on features defined on the transitions/moves.

This paper will explore how we can improve the accuracy of the arc-standard baseline system that was implemented during Lab three and four by adding new features. Studies have shown that richer feature sets can improve the attachment score significantly in an arc-eager process, how the accuracy behaves in an arc-standard process has not been studied alone.

It is interesting to continue study transition-based dependency parsing since the language structure is imperfect and the meaning a sentence is interpreted contextually. This context based interpretation reveals important information for a NLP-system. An example of this is: "He is driving into the garage in his new car." This is naturally interpreted as someone sitting in a car and driving into a garage. Another non-contextual though grammatically correct interpretation could be that someone drives into a garage which is located inside a new car. This can be applied in many systems, such as language translation, text generation, sentiment analysis etc.

1.1 The baseline system

The baseline system that was implemented during the labs consists of two parts, a static part that implements the logic of the arc-standard algorithm and a non-static part that includes a fixed-window model as the learning component.

1.1.1 The arc-standard algorithm

The arc-standard algorithm is an algorithm used for transition based dependency parsing that only can predict projective dependency trees, meaning that every subtree match a contiguous sequence of words. This means that we can only define three transitions for the configuration of an arc-standard algorithm. The configuration for a parser has three parts, a buffer containing words in a sentence that are yet to be processed, a stack containing the words that are being processed at the moment and lastly a partial dependency tree containing the relations between the processed words. The mentioned transitions for the arc-standard algorithm are as follows.

- Shift transition: Places the buffer's frontmost word on top of the stack.
- Left-arc transition: Creates a dependency relation between the frontmost word of the stack to the word below and then removes the word below from the stack.

- Right-arc transition: Creates a dependency relation between the second-topmost word of the stack to the word at the top and then removes the topmost word from the stack.

1.1.2 Baseline features

The baseline uses 6 features, of which all are word-embeddings. The words and tags used in the features are as follows.

1. The first word in the buffer.
2. The topmost word of the stack.
3. The second topmost word of the stack.
4. The first tag in the buffer.
5. The topmost tag of the stack.
6. The second topmost tag of the stack.

1.1.3 The fixed-window model

The learning component for the baseline system is the fixed-window model. The fixed-window model has the following architecture, the input to the network is a vector with all features, the features are mapped to an embedding vector and are then concatenated to a vector. The concatenated vector is then passed through a feed-forward network.

1.1.4 Evaluation metric

The parser is evaluated with Unlabeled Attachment Score (UAS) which the amount of correctly predicted heads divided by the total amount of data points.

2 Literature review

The transition based dependency parsing uses deterministic shift-reduce process for making structural predictions. It's different from the graph based dependency parsing and offers linear time complexity and the possibility to define non local features.

In the paper, Zhang and Nivre apply beam search and partial dynamic programming, in order to improve the results obtained with the greedy one best search, which is the default search implemented in the baseline. Moreover, in training, they use global structural learning (to replace local learning) on each decision.

But the main focus of this paper is another aspect: the feature definition. It represents the type of information that are used by a statistical system to make his predictions. Therefore, it could be a major factor when it comes to determine parsing accuracy. Most of the time, transition-based dependency parsers use only a few non-local features, but this paper aims at exploring richer feature representation and show how they can improve parsing accuracy. Last but no least, the projective dependency trees can be built using arc-standard or arc-eager process. This modifies the actions that are possible to do at each step. The baseline we will implement in lab 3 and 4 uses the arc-standard process, while Zhang and Nivre adopted the arc-eager process. Zhang and Nivre define some new variables. S_0 which is the top of the stack and N_0 the front item of the queue. They also define define the head of S_0 (if any) with S_{0h} , the leftmost and rightmost modifiers of S_0 with S_{0l} and S_{0r}. In the same manner N_{0l} is defined. The feature templates implemented in this work are the following:

- Distance between S_0 and N_0 (compatible with arc-standard process)

- Valency of N_0 and S_0 (not compatible with arc-standard process)
- Unigram information for S_{0h}, S_{0l}, S_{0r}, and N_{0l} (compatible with arc-standard process)
- Third-order features of S_0 and N_0 (not compatible if we don't change definition of right arc action)
- Set of dependency labels with S_0 and N_0

Nonetheless all this features are not compatible with the arc-standard algorithm. In order to test some of them we will change their definition. The arc-eager algorithm allows to access and tag the word in the top of the queue (here it is N_0) but this not possible with the arc-standard. So we will have to change N_0 and consider S_1, the second element in the stack, instead.

3 Task assignment

The aim of this project is divide the workload between the members evenly and to do so it is important to identify bottle necks. One such bottle neck can be caused by features requiring different amounts of information from the parser which can be avoided. The first step of the project is therefore to develop a baseline that is compatible with all the new features. From there the features can be worked on, tested and evaluated separately. When all features are tested and evaluated, a model containing all features can be implemented and evaluated.

4 References

Zhang, Y. and Nivre, J. (2011) Transition-based Dependency Parsing with Rich Non-local Features. [pdf] Association for Computational Linguistics. Available at: https://www.aclweb.org/anthology/P11-2033.pdf [Accessed 14 February 2021].