IT412: Natural Language Processing

Assignment 6: Dependency Parser

Instructor: Prasenjit Majumder

Learning Outcome: At the end of this assignment you will creating a dependency parser using BiLSTM. And evaluating it using accuracy

1 Problem description

Dependency parsing is the process of analyzing the grammatical structure of a sentence based on the dependencies between the words in a sentence. In Dependency parsing, various tags represent the relationship between two words in a sentence. These tags are the dependency tags. A dependency involves only two words in which one acts as the head and other acts as the child. Example of dependency between words in a sentence is shown below

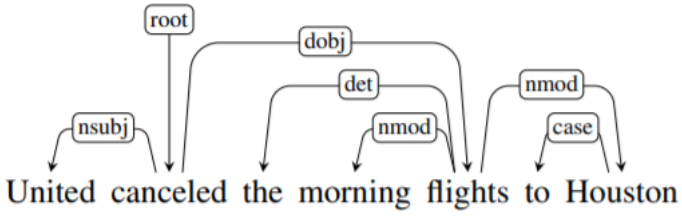

Figure 1: Example of dependency between words in a sentence

1.1 Transition Based Dependency Parser

Transition based dependency parser is a greedy approach for implementing a dependency parser. It consists of a Stack, Buffer and Oracle as shown in Figure 2. Stack contains the words whose relation we need to find. Buffer contains words from the sentence whose relation has not been found. Oracle is a classifier which predicts the dependency relation.

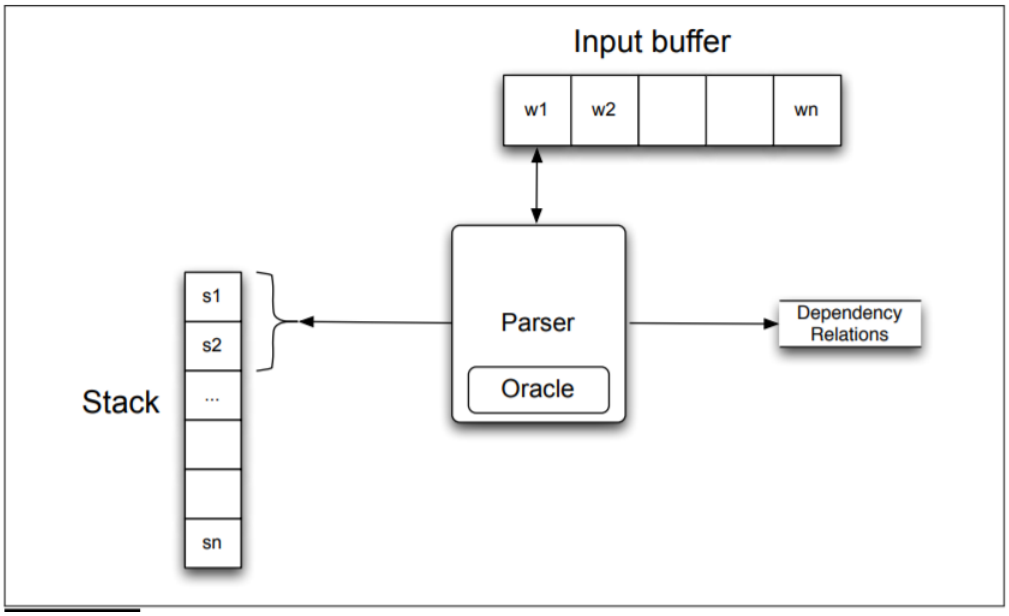

Figure 2: Transition Based Parser

It consists of three operations:

- Left Arc: Arc from word at top of stack and word directly within the top word of the stack. Remove lower word from stack
- Right Arc: Arc from 2nd word in stack to word at top of stack. Remove top word from stack
- Shift: Remove word from front of input buffer and push it into the stack.

2 Implementation

2.1 Dataset

- The dataset is available in the github link: https://github.com/rasoolims/nlp_hw_dep
- Clone it and run the following commands:
  - python src/gen vocab.py trees/train.conll data/vocabs
  - python src/gen.py trees/train.conll data/train.data
  - python src/gen.py trees/dev.conll data/dev.data
  - python src/gen.py trees/test.conll data/test.data
- data/vocabs.word: This file contains indices for words. Remember that <null> is not a real word but since there are some cases that a word feature is null, we also learn embedding for a null word. Note that <root> is a word feature for the root token. Any word that does not occur in this vocabulary should be mapped to the value corresponding to the <unk> word in this file.
- data/vocabs.pos: Similar to the word vocabulary but this is for partof-speech tags. There is no notion of unknown in POS features.
- data/vocabs.labels: Similar to the word vocabulary but this is for dependency labels. There is no notion of unknown in dependency label features.
- data/vocabs.actions: This is a string to index conversion for the actions(this should be used for the output layer).
- In train, dev and test each line shows one data instance. The first 20 columns are for wordbased features. The second 20 columns are for the POS-based features. The next 12 features are for dependency labels. The last column shows the action. Remember that, when you want to use these features in implementation, you should convert them to integer values, based the vocabularies generated.

2.2 Exercise

- Create a stack, buffer. Initialize the stack with ROOT.
- Represent the word using concatenation of word vector and POS tag embedding
- Pass the words through BiLSTM. Each word will be represented using the concatenation of forward LSTM and backward LSTM
- Use the concatenation of 3 words at top of the stack and 1 word from buffer and pass it through a fully connected neural network, apply softmax on top to predict the operation.
- Use the train set for training, for fine tuning the hyperparameters use the dev set. And for testing use test set.
- In this assignment we are not predicting the dependency relation tag, we are only predicting the left arc or right arc so the evaluation metric here is Unlabeled Accuracy Score (UAS).

3 References

- https://gzwq.github.io/2019/06/12/NLP-Dependency-Parser/
- https://web.stanford.edu/~jurafsky/slp3/14.pdf
- https://pytorch.org/docs/stable/data.html
- Kiperwasser, Eliyahu, and Yoav Goldberg. "Simple and accurate dependency parsing using bidirectional LSTM feature representations." Transactions of the Association for Computational Linguistics 4 (2016): 313-327.

4 Submission

- You have to submit your assignment in notebook with proper comments and explanation of your approach.
- Report the accuracy for the approach
- The submission deadline for this assignment is **4th October 2021 11 pm**