Assignment 3: Dependency parsing

August 22, 2018

In this assignment we shall learn dependency parsing using NLTK library.

Question 1: Learning a dependency parser from a treebank

In this question we shall learn to train and evaluate transition-based dependency parser.

- Use the train and test splits of the English universal dependency treebank given in *UD_Hindi.zip*.
- Use the functions given in the nltk.parse.transitionparser for this assignment.
- The training data is in 'conllu' format.

The steps to be followed are as follows;

- **Training**
  - Read the CONLLU data into dependency graphs.
  - Write the configuration states and corresponding transitions into the training file as required by the learning algorithm in sklearn.
  - Train a parser using the this file.
- **Testing**
  - Test the trained model on the test data. For evaluation use the library **nltk.parse.evaluate** and report the results in terms of labelled and unlabelled attachment scores.

File format:

- The train and test files contain 500 and 100 parse trees respectively.
- The Hindi words and lemma are in wx format.
- The 'CONLLU' format in the file is slightly different from the one used by NLTK. Please check the formats and make necessary changes.
- The 10^{th} column contains some additional features that you may use as morphological feature or ignore.

Features: The NLTK system by default uses the word, PoS, lemma and morphological features.

- Compare the performance of the parser using different combinations of features (particularly with and without morphological features).
- You may also try to use some features in the 10^{th} column.

In the report clearly state how you have removed or introduced some features. Also submit the corresponding codes.

Repeat the experiments for the following cases

- Arc-eager transition.
- Arc-standard transitions.

NLTK trains a svm classifier to learn the predict the transition corresponding to a configuration. For each feature combination and transition scheme (eager or standard) report the results with

- *logistic regression*
- *MLP classifier.*

Deliverables

- Python code in .ipynb notebook.
- A detailed README to run the code (Separate from report. A test file will do.)
- A report containing *the results of the experiments, your analysis and conclusion* in pdf format. (Please don't insert your code in the report)