Prof. Björn Gambäck
TA. Tollef Jørgensen

Lab 4 – Grammars and parsing

Deadline: 23:59, March 16th, 2022

Guidelines

This lab is somewhat similar to the last one in terms of setup. You are to write functions based on descriptions, and sometimes fill in missing pieces. Code is available as .py and .ipynb. Using the latter (Jupyter Notebook) is highly recommended, as it provides substantially better feedback throughout the lab. All information for this exercise can be found in the appended code (which can also be accessed on GitHub[1]).

Provide your outputs in a simple report, along with textual answers (remember to justify your choices and note down any interesting observations or issues). Zip if there's more than 1 file.

As always, follow the format first_last_lab4.zip/ipynb.

Exercise 1 - Introduction to chunking

1. Make your own noun phrase (NP) chunker, detecting noun phrases and a clause, for which verbs (VB) are followed by a preposition (IN) and/or a noun phrase. This requires you to define grammar and a regex based chunk parser.
2. Convert a POS tagged text to a list of tuples, where each tuple consists of a verb followed by a sequence of noun phrases and prepositions (i.e., the *clause* above) Example: "the little cat sat on the mat" becomes ('sat', 'on', 'NP') . . .
3. Using the pre-annotated test set from wall street journal data (conll2000 in nltk), experiment with different grammars to get the highest possible F-measure. There is no evaluation of this task, but rather a motivator to learn something about grammars.

Exercise 2 - Making use of chunks

1. With the following grammar rules:

   a) proper noun singular

   b) determiner followed by an adjective

   c) two consecutive nouns

   Create a *RegexpParser* chunker.
2. Read the file 'starlink.txt' and perform the following operations on the text: sentence tokenize, word tokenize, and pos tag. Finish the described functions in the code.
3. From all found subtrees in the text, print out the text from all the leaves on the form of 'DT -> JJ' (defined above)
4. Create a custom rule for a combination of 3 or more tags, similarly to task c).

   Do you see any practical uses for chunking with this rule, or in general?

[1] `https://github.com/ph10m/TDT4310-Exercises`

Prof. Björn Gambäck
TA. Tollef Jørgensen

Exercise 3 - Context-free grammar (CFG)

1. Create a cfg to handle sentences such as "she is programming", "they are coding" (look at the word forms and POS). The first verb should only handle present tense, while the second verb is flexible. Note that you need to specify the accepted words.
2. Find some problems with the CFG you just defined, any ideas how you would improve the results?
3. Initialize a 'ChartParser' with the cfg from 3a). Write a function to retrieve words not defined by your grammar.
4. Output the tree of your parser for the sentence "they are programming"

Exercise 4 - Tweet like Trump! Now that he's banned

Using the provided file "realDonaldTrump.json", you will build a language model to generate Trump-esque tweets using n-grams. Hint: make use of "padded_everygram_pipeline" supported in nltk.lm. This creates all ngrams up to the specified N-parameter with padding.

Then, create a grammar to chunk some typical trump statements.

There are multiple approaches to this. One way would be to use your own input to the model and look at the resulting outputs and their POS tags. Another possible approach is to use the training data to group together e.g. 5-grams of POS tags to look at the most frequently occurring POS tag groupings. The aim is to have a chunker that groups utterances like "so sad", "make america great again!" and so forth.

Show your results using the outputs from your model with these inputs:

- clinton will
- obama is
- build a
- so sad