COMP 6751

Natural Language Processing: Project 2

Developing a context free feature grammar

Submitted To: Dr. Sabine

Submission Date: October 14 2023

Iymen Abdella

Student No. 40218280

IMPROVEMENTS FROM PREVIOUS PROJECT

Several improvements were made to the codebase during the second project phase to better accommodate future features that will need to be integrated into the pipe and filter design. The pipe and filter design for text processing is a simple and scalable solution, but some refactoring had to be done to leverage the design pattern to its fullest potential. Below are some key improvements to the codebase to encourage better flexibility and adaptability.

UNIT GAZETTEER

The Unit Gazetteer has been updated to include a wider range of acceptable units with annotated POS tags for each unit. This improvement should help identify gazetteer members better and allow for additional analysis for POS tagging. Additionally, the gazetteer is much larger than the previous one from PINT, with an estimated ~1500 entries compared to the prior ~700.

As well the gazetteer has been decoupled from the main pipeline to allow for better modularity and simpler integration with the main flow of execution. Many gazetteers can be queried at the same time, without a drastic impact to overall performance.

PIPE AND FILTER DESIGN

As previously mentioned, considerable time has been spent to refactor the codebase to favor a pipe and filter design pattern to yield a more modular and reusable processing pipeline. Gazetteers, grammars, and entity detectors have been decoupled from the main processing pipeline to allow for better scalability and increased performance while keeping the codebase simple and easy to understand.

The dividends are expected to be yielded in the coming weeks as more features will be added to the processing pipeline. With this refactored design new features can be easily integrated into the existing feature set without drastically impacting performance or requiring substantial changes from the current implementation.

ENTITY DETECTION

The entity detection has been expanded into its own standalone module, which can scale better to detect a wider variety of features. Currently the module contains support for non-binary named entity detection, as well as a more robust measured entity detector. For future features, adding a new entity detector will prove much simpler with the introduction of the *entityDetector.py* module.

IMPROVED USER INTERACTION

Along with the improvements previously described, the user interaction with the pipeline has been improved to be simpler to understand and offer more options for analyzing texts. Every option can be repeated as many times as the user wishes, leading to improved ease of use and smoother testing/validation. There are three main methods for analyzing texts:

- **Direct command line input (i)**: the user can simply copy a legible sentence directly to the command line.
- **Analyze a reuters corpus document :** the user can also copy a file ID from the reuters corpus directly on the command line for analysis.
- **Validate many texts at once (v):** the user also has the ability to analyze all the texts stored under the *validation_Text* folder at once.

All options permit the user to have the results displayed on the command line and/or stored locally for later analysis. With these improvements the ease of use has been dramatically improved, and will lead to better validation, analysis and testing options depending on the user's needs.

TEXT ANNOTATION

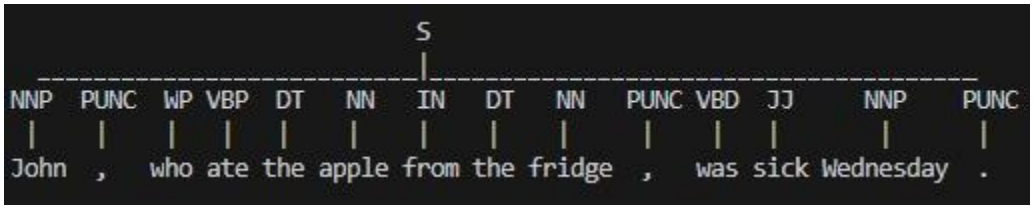
We have leveraged the Pandas library for simple storage and management of annotated text in CoNLL format. Some features such as POS and NE are features taken directly from the NLTK module without any modifications. However, the entity detection has been improved for better coverage, and formatting to conform to the CoNLL layout.

CFG DESIGN

To construct a Context Free grammar that would sufficiently cover the validation data we started with a POS analysis of the sentences. Then we used the sentences' POS layout as the first layer of productions which would propagate from the start symbol.

Afterwards, we created a bottom layer of productions which mapped the POS terms directly to the words themselves. With these two constructions in our CFG we can sufficiently cover all the validation data, however the coverage is rigid and specific to the sentences that served as the foundation.

It is elementary to observe the inflexible nature once a sentence is parsed through our CFG and the resulting tree structure is extremely flat:



Therefore, we supplemented the CFG with some complements that offer more flexibility and leverage the subcategories present within some of the sentences' constituents. Once validated we updated our grammar with the LHS of the complements so they can be denoted by the parser.

The subcategories considered were:

- Noun phrases: capturing plural and singular nouns, as well as determiners before nouns.
- Verb Phrases: capturing adjectives after verbs, noun phrases following verbs, and different types of verb tenses.
- Cardinal Phrases: capturing nouns following numbers, including any intermediate numbers.
- Prepositional phrases: capturing prepositional phrases or prepositions preceding a noun phrase.

RESULTS

As captured in the associated demo.pdf, the tree output has a lot more depth, however the parser does reveal different possible trees depending on the sentence structure.

Some sentences have multiple possible trees because there are multiple interpretations based on the Context free grammar that are valid. So the user will find the different tree structures within the output.

For future improvements we would simplify our rules to reduce the possible number of interpretations as well as implement more robust rules for better coverage and improved tree balance.