COMP 6751 Natural Language Analysis

Project Report 2:

Developing Context Free Grammar

Student name: Md Sakib Ullah Sourav

ID: 40264066

Expectations of originality:

I, Md Sakib Ullah Sourav (student id 40264066), certify that this submission is my original work and meets the Faculty's Expectations of Originality.

Date: October 12, 2023

Contents

1. Project Goal

As described in the project description, I developed a context-free grammar (CFG) that handles a variety of textual content, and I also generated a parse tree to show the syntactic structure using NLTK (Natural Language Toolkit). Subsequently, in the later part of the project, the outputs of the CFG such as- Parts of speech (POS) tagging, Named Entity (NE), Entity (Ent) and Measured Entity (MeasEnt) have been shown in CoNLL (Conference on Computational Natural Language Learning) format.

2. Methods

Below I listed the methods I followed to accomplish the project goal-

2.1 EarleyChartParser

With its ability to handle a broad variety of context-free grammars, the EarleyChartParser parsing algorithm is very adaptable and may be used for a wide range of natural language processing applications, including syntactic parsing and semantic analysis. This tool is highly advantageous in examining the composition of sentences in natural language and may be tailored to fit a range of languages and purposes. It is a chart parsing algorithm named for its author, Jay Earley [1].

Similar to top-down statement parsing, the Earley algorithm determines whether a leaf contains a certain part of speech based on a lexicon. Encoded as a dictionary, this lexicon associates every word with a list of possible components of speech.

2.1.2. nltk.tree.prettyprinter

A parse tree can be formatted and made more comprehensible for humans by using the TreePrettyPrinter class in NLTK, which is specifically a component of the tree module. It provides techniques to produce a well formatted textual representation of a parse tree given an input parse tree. This is especially useful for those who wish to see and examine a sentence's or text's syntactic structure.

2.1.3 spaCy

For this project, I displayed the CFG output in CoNLL format using spaCy. SpaCy is a free and open-source framework for natural language processing that is intended to process text data quickly and effectively. Tokenization, part-of-speech tagging, named entity recognition, dependency parsing, and other Natural Language Processing (NLP) tasks are among the many applications for which it is commonly employed.

Pre-trained models for several languages, including English, are offered by spaCy. These models are trained on extensive text datasets and have a range of capabilities.

2.1.4 en_core_web_sm

I used en_core_web_sm as the backbone of spaCy, which is one of the pre-trained models for English provided by spaCy. It's a relatively small and efficient model designed for general-purpose NLP tasks in English. NLTK is more of a toolkit and library for NLP, while spaCy is a library and framework that focuses on efficiency and production-level processing.

3. Implementations

To accomplish the task of this project, first I had to develop a context-free language that addresses the text validation content. Generally speaking, it is convenient to save the rules for editing, testing, and revision in a file when developing a grammar.

3.1 Grammar Tagset Generation

For the list of words corresponding to the parts of speech, I developed a tagset from Penn treebank that has been placed at the first cell in my code. The grammatical word rules that I inserted in the tagset are given below-

```
tagset = {
  # Coordinating conjunction
  "CC": ["but", "and", "that", "against", "both", "which", "or"],

  # Cardinal number
  "CD": [],

  # Determiner
  "DT": ["a", "an", "the"],

  # Existential there
  "EX": [],

  # Foreign word
  "FW": [],

  # Preposition or subordinating conjunction
  "IN": ["in", "on", "at", "from", "in", "on", "at", "from", "to", "with",
"against", "for", "of"],

  # Adjective
  "JJ": ["healthy", "sick", "happy", "sad", "angry", "productive", "lazy",
"his", "her", "their", "your", "last", "promising", "that", "those",
"crunchy", "both", "semantic", "deep", "dutch", "several", "some",
"further", "logical", "two"],

  # Adjective, comparative
  "JJR": ["healthier", "sicker", "happier", "sadder", "angrier", "lazier",
"crunchier", "deeper", "90"],

  # Adjective, superlative
  "JJS": ["healthiest", "sickest", "happiest", "saddest", "angriest",
"laziest", "crunchiest", "deepest"],

  # List item marker
  "LS": [],

  # Modal
  "MD": [],

  # Noun, singular or mass
  "NN": ["apple", "PhD", "mango", "orange", "banana", "heart", "rate",
"beats/minute", "breaths/minute", "/microliters", "mmhg", "leukocyte",
"count", "immature", "forms", "bands", "body", "temperature", "degrees",
"celsius","respiratory", "rate", "partial", "pressure", "co2", "grape",
"table", "chair", "bat", "ball", "fridge", "office", "morning", "flight",
"yesterday", "today", "tomorrow", "refrigerator", "week", "last", "will",
"desk", "colleague", "replacement", "day", "everyday", "thesis",
"representation", "representations", "parser", "sentences", "issues",
"improvements", "system", "generator", "style", "form", "inference",
"two", "ways"],

  # Noun, plural
  "NNS": [],

  # Proper noun, singular
  "NNP": ["Sakib", "Huda", "John", "Sue", "James", "Robert", "Dutch",
"Mary", "Patricia", "Jennifer", "O-Malley", "Monday", "Tuesday",
"Wednesday", "Thursday", "Friday", "Saturday", "Sunday", "Janurary",
"Febuary", "March", "April", "May", "June", "July", "August", "September",
"October", "November", "December", "Delilah"],

  # Proper noun, plural
  "NNPS": [],

  # Predeterminer
  "PDT": [],
  # Possessive ending
  "POS": [],

  # Personal pronoun
  "PRP": ["I", "it", "he", "she", "they", "that" ,"them", "both", "this",
"some", "several"],

  # Possessive pronoun
  "PRPS": [],

  # Adverb
  "RB": ["not", "finally", "last", "some", "further", "mainly", "two",
"deep"],

  # Adverb, comparative
  "RBR": [],

  # Adverb, superlative
  "RBS": [],

  # Particle
  "RP": [],

  # Symbol
  "SYM": [],

  # to
  "TO": [],

  # Interjection
  "UH": [],

  # Verb, base form
  "VB": ["eat", "prefer", "be", "last", "over", "under", "take",
"greater", "than", "less", "than", "promise", "will", "put", "intend",
"share", "anticipate", "treat", "delight", "say", "focus", "parse"],

  # Verb, past tense
  "VBD": ["ate", "preferred", "was", "were", "lasted", "took", "promised",
"would", "put", "intended", "shared", "anticipated", "treated",
"delighted", "said", "focussed", "focused", "parsed", "developed",
"optimized"],

  # Verb, gerund or present participle
  "VBG": ["eating", "preferring", "being", "lasting", "taking",
"promising", "putting", "intending", "sharing", "anticipating",
"treating", "delighting", "saying", "focussing", "focusing", "parsing",
"discussing", "proposing"],

  # Verb, past participle
  "VBN": ["eaten", "preferred", "been", "lasted", "taken"],

  # Verb, non-3rd person singular present
  "VBP": ["am", "are", "will"],

  # Verb, 3rd person singular present
  "VBZ": ["is", "eats", "prefers", "lasts", "takes", "promises",
"intends", "shares", "anticipates", "treats", "delights", "says",
"focuses", "parses", "describes", "computes"],

  # Wh-determiner
  "WDT": [],

  # Wh-pronoun
  "WP": ["who", "which"],

  #   Possessive wh-pronoun
  "WPS": [],

  # Wh-adverb
  "WRB": [],

}

# Including numbers 0 to 100 as common nouns
[tagset["NN"].append(str(x)) for x in range(101)]

# Including numbers 1000 to 20230 as common nouns
[tagset["NN"].append("20"+str(x)) for x in range(24)]
```

Then, an Earley chart parser in NLTK is used to parse the developed CFG grammars.

3.2 Context Free Grammar (CFG) Generation

In the second corresponding code cell, I inserted the developed CFG grammar like below-

""""

```
S -> NP | VP | PP | NP VP | VP NP | NP ',' NP VP | NP ',' NP VP ',' VP | PP ',' S | S CC S| S ',' CC S | S PP | NP ',' PP ',' NP VP NP
    PP | S ',' V S

NP -> ProN | PropN | N | DT NP | CC NP | ADJ NP | RB NP | N NP | PropN NP | ProN NP | NP PP

ProN -> PRP | PRPS | WP | WPS

PropN -> NNP | NNPS

N -> NN | NNS


VP -> V | V NP | V VP | V NP PP | V PP | V ADJ | V ADJ NP | RB VP | VP PP

V -> VB | VBD | VBG | VBN | VBP | VBZ

PP -> IN NP | IN VP | PP VP


ADJ -> JJ | JJR | JJS

"""
```

The above CFG is developed by drawing inspiration from the NLTK book in [2].

3.3 CGF Outputs in CoNLL format

Next, I adopted spaCy library [4] to demonstrate the POS, NE, Ent and MeasEnt in CoNLL format. Linguistic data can be shared and annotated in a tabular manner using the CoNLL format. A word or token is represented by each row in the table, and other details about those tokens are contained in the columns.

To display POS as output I used-

```
pos = token.pos_
```

For NE output,

```
ne = token.ent_type_
```

In order to show Ent and MeasEnt as output, I used the below commands correspondingly,

```
ent = "Ent" # Entity Type (customize based on your data)
meas_ent = "MeasEnt" # Measured Entity (customize based on your data)
```

At the end, to show all the CFG outputs in CoNLL format, I wrote the below code-

```
    print(f"{token_index}\t{word}\t_\t{pos}\t{ne}\t{ent}\t{meas_ent}")
```

4. Limitations

Please refer to the Demo file.

5. Reference

[1] Þorsteinsson, V., Óladóttir, H., & Loftsson, H. (2019, September). A wide-coverage context-free grammar for Icelandic and an accompanying parsing system. In Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019) (pp. 1397-1404). doi: https://aclanthology.org/R19-1160.pdf

[2] Bird, Steven, Ewan Klein, and Edward Loper. "Analyzing Text with the Natural Language Toolkit." Natural Language Processing with Python. O'Reilly Media, Inc, 2009. 504. doi: https://www.nltk.org/book/

[3] NLTK. (2023). https://www.nltk.org/py-modindex.html

[4] Spacy · industrial-strength natural language processing in python. · Industrial-strength Natural Language Processing in Python. (2023). https://spacy.io/