

Teacher Assessment 2

Programming Assignment

COURSE:

Language Processors (CST412)

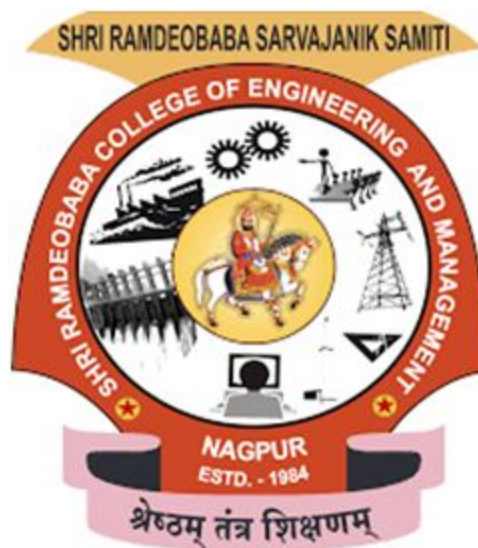
Group No. 16

Ananya - 39

Vedant Khairnar - 71

Kheever Choudhary - 47

Shrijeet Shivdekar - 61



Introduction to Natural Language Processing for Text

What is NLP (Natural Language Processing)?

NLP is a subfield of computer science and artificial intelligence concerned with interactions between computers and human (natural) languages. It is used to apply **machine learning** algorithms to **text** and **speech**.

For example, we can use NLP to create systems like **speech recognition, document summarization, machine translation, spam detection, named entity recognition, question answering, autocomplete, predictive typing** and so on.

Nowadays, most of us have smartphones that have speech recognition. These smartphones use NLP to understand what is said. Also, many people use laptops whose operating system has built-in speech recognition.

Introduction to the NLTK library for Python

NLTK (**Natural Language Toolkit**) is a **leading platform** for building Python programs to work with **human language data**. It provides easy-to-use interfaces to **many [corpora](#) and lexical resources**. Also, it contains a suite of **text processing libraries** for classification, tokenization, stemming, tagging, parsing, and semantic reasoning. Best of all, NLTK is a free, open source, community-driven project.

We'll use this toolkit to show some basics of the natural language processing field. For the examples below, I'll assume that we have imported the NLTK toolkit. We can do this like this:

```
import nltk.
```

The Basics of NLP for Text

In this article, we'll cover the following topics:

1. Sentence Tokenization
2. Word Tokenization
3. Text Lemmatization and Stemming
4. Stop Words
5. Regex
6. Bag-of-Words
7. TF-IDF

The phases of NLP

There are 5 phases of NLP

1. Morphological Analysis/ lexical Analysis
2. Syntax Analysis
3. Semantic Analysis
4. Discourse integration
5. Pragmatic Analysis

Lexical Analysis

Lexical analysis takes the modified source code from language preprocessors that are written in the form of sentences. The lexical analyzer breaks these syntaxes into a series of tokens, by removing any whitespace or comments in the source code.

The lexical analysis in NLP deals with the study at the level of words with respect to their lexical meaning and part-of-speech. This level of linguistic processing utilizes a language's lexicon, which is a collection of individual lexemes. A lexeme is a basic unit of lexical meaning; which is an abstract unit of morphological analysis that represents the set of forms or "senses" taken by a single morpheme.

"Duck", for example, can take the form of a noun or a verb but its part-of-speech and lexical meaning can only be derived in context with other words used in the phrase/sentence. This, in fact, is an early step towards a more sophisticated Information Retrieval system where precision is improved through part-of-speech tagging.

Syntax Analysis in NLP

Syntactic analysis or parsing or syntax analysis is the third phase of NLP. The purpose of this phase is to draw exact meaning, or you can say dictionary meaning from the text.

Syntactic Analysis is used to check grammar, word arrangements, and shows the relationship among the words.

Example: Agra goes to the Poonam

In the real world, Agra goes to the Poonam, does not make any sense, so this sentence is rejected by the Syntactic analyzer.

Discourse Integration

In any language, we find dependencies like pronouns, its work is to identify and understand where the dependency is pointing to.

Example

Ram is a boy, he goes to school

Here "he" is a dependency pointing to Ram.

Pragmatic Analysis

It is said to be the toughest part in AI, pragmatic analysis deals with the context of a sentence.

For example: John saw Mary in a garden with a cat- here we can't say that John is with a cat or Mary is with a cat. Bonus Example(for a little laugh): Do you know what time is it? One context of this is asking what time it was, another might be an Indian mom yelling at son to make him wake up, definitely, she is not asking you time there.

Semantic Analysis

The aim of semantic analysis is to give the machine the ability to understand human language. The role of the analysis part is to give exact meaning of text and to check whether grammar is correct or not. The role of analyzer is to check text for meaningfulness. There are two important parts in semantic analysis- lexical semantics and combination of words.

Lexical semantics deal with meanings of individual words. There is a common misconception that combining words' meaning would give us a complete understanding of the sentence.

The combination of words' phase has 3 basic parts - relationship between words, structure of sentence and real world knowledge. The characteristics of this phase include deciphering how words combine to form larger meaning, dealing with form or structure of constriction of sentence and understanding the meaning of each word. The classic example - I like you and you like me- they tell us how the same words combine in a different way to give different meanings.

The basic building blocks of semantic analysis are- entities, concept, relations and predicates. Entities represent individuals. For eg: Haryana. Concepts represent general categories and are analogous to common nouns. For example, person, vehicle, state. Relations describe the relationship between entities and concepts. For example, Haryana is a state. Predicates are the entities which represent verb structure, for example I like you.

The approaches to semantic analysis include FOPL, semantic nets and case grammar. First order predicate logic(FOPL) is used to convert sentences to predicate logic. It is easy for machines to understand. Eg: Jack Loves Jill is love(Jack, Jill). Semantic nets give semantic relation between concepts. Case grammar gives analysis of structure of the sentence. Eg: we gave our dog a bone: We= subject, gave= verb, dog= Indirect object, bone= direct object.

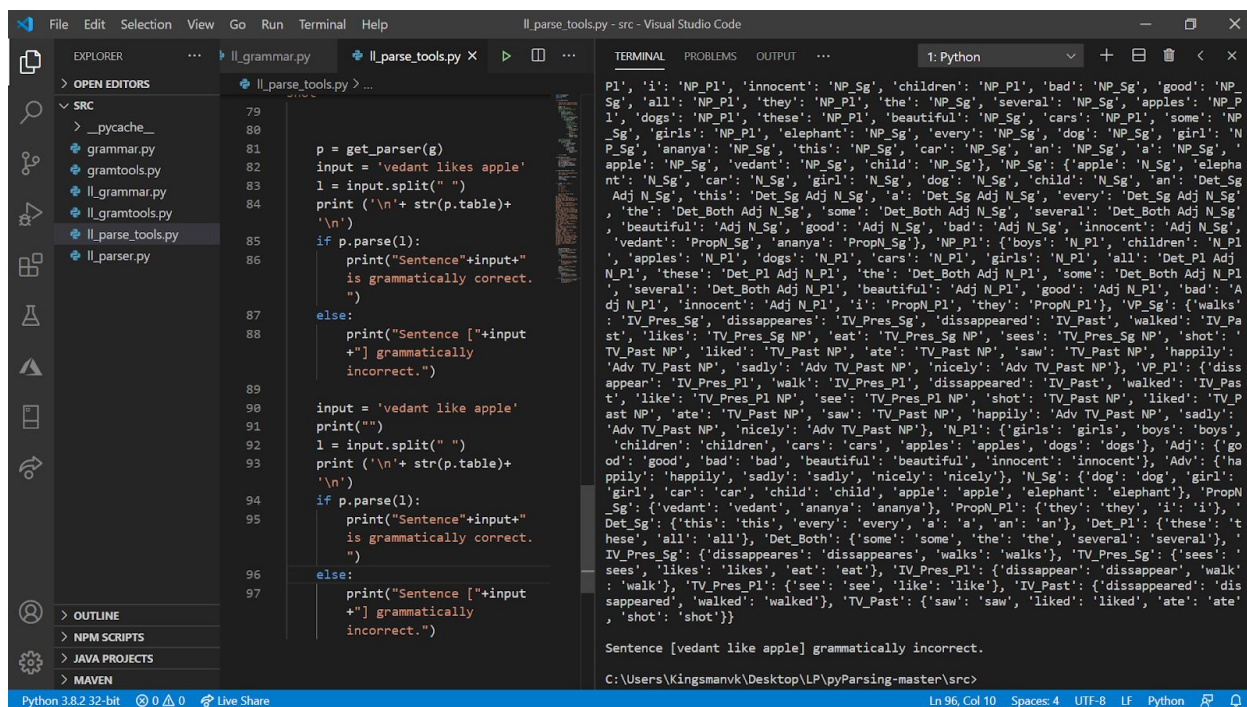
Programs:

The code is in the zip file attached.

Github Repository: <https://github.com/VedantKhairnar/NLP-LL1-Parser>

Output:

Input: vedant like apple



```
File Edit Selection View Go Run Terminal Help
ll_parse_tools.py - src - Visual Studio Code

EXPLORER
> OPEN EDITORS
  ll_parse_tools.py > ...
> SRC
  > __pycache__
  grammar.py
  gramtools.py
  ll_grammar.py
  ll_gramtools.py
  ll_parse_tools.py
  ll_parser.py
> OUTLINE
> NPM SCRIPTS
> JAVA PROJECTS
> MAVEN

ll_parse_tools.py
79
80
81 p = get_parser(g)
82 input = 'vedant likes apple'
83 l = input.split(" ")
84 print ('\n'+ str(p.table)+
85       '\n')
86 if p.parse(1):
87     print("Sentence"+input+"
88         is grammatically correct.
89     ")
90 else:
91     print("Sentence ["+input
92         +"] grammatically
93     incorrect.")
94
95 input = 'vedant like apple'
96 print("")
97 l = input.split(" ")
98 print ('\n'+ str(p.table)+
99       '\n')
100 if p.parse(1):
101     print("Sentence"+input+"
102         is grammatically correct.
103     ")
104 else:
105     print("Sentence ["+input
106         +"] grammatically
107     incorrect.")

TERMINAL
1: Python
P1', 'i': 'NP_P1', 'innocent': 'NP_Sg', 'children': 'NP_P1', 'bad': 'NP_Sg', 'good': 'NP_
Sg', 'all': 'NP_P1', 'they': 'NP_P1', 'the': 'NP_Sg', 'several': 'NP_Sg', 'apples': 'NP_P
1', 'dogs': 'NP_P1', 'these': 'NP_P1', 'beautiful': 'NP_Sg', 'cars': 'NP_P1', 'some': 'NP
_Sg', 'girls': 'NP_P1', 'elephant': 'NP_Sg', 'every': 'NP_Sg', 'dog': 'NP_Sg', 'girl': 'N
P_Sg', 'ananya': 'NP_Sg', 'this': 'NP_Sg', 'car': 'NP_Sg', 'an': 'NP_Sg', 'a': 'NP_Sg', '
apple': 'NP_Sg', 'vedant': 'NP_Sg', 'child': 'NP_Sg', 'NP_Sg': {'apple': 'N_Sg', 'elepha
nt': 'N_Sg', 'car': 'N_Sg', 'girl': 'N_Sg', 'dog': 'N_Sg', 'child': 'N_Sg', 'an': 'Det_Sg
Adj N_Sg', 'this': 'Det_Sg Adj N_Sg', 'a': 'Det_Sg Adj N_Sg', 'every': 'Det_Sg Adj N_Sg'
, 'the': 'Det_Both Adj N_Sg', 'some': 'Det_Both Adj N_Sg', 'several': 'Det_Both Adj N_Sg'
, 'beautiful': 'Adj N_Sg', 'good': 'Adj N_Sg', 'bad': 'Adj N_Sg', 'innocent': 'Adj N_Sg'
, 'vedant': 'PropN_Sg', 'ananya': 'PropN_Sg'}, 'NP_P1': {'boys': 'N_P1', 'children': 'N_P1'
, 'apples': 'N_P1', 'dogs': 'N_P1', 'cars': 'N_P1', 'girls': 'N_P1', 'all': 'Det_P1 Adj
N_P1', 'these': 'Det_P1 Adj N_P1', 'the': 'Det_Both Adj N_P1', 'some': 'Det_Both Adj N_P1'
, 'several': 'Det_Both Adj N_P1', 'beautiful': 'Adj N_P1', 'good': 'Adj N_P1', 'bad': 'A
dj N_P1', 'innocent': 'Adj N_P1', 'i': 'PropN_P1', 'they': 'PropN_P1', 'VP_Sg': {'walks'
: 'IV_Pres_Sg', 'disappeares': 'IV_Pres_Sg', 'disappeared': 'IV_Past', 'walked': 'IV_Pa
st', 'likes': 'TV_Pres_Sg NP', 'eat': 'TV_Pres_Sg NP', 'sees': 'TV_Pres_Sg NP', 'shot': '
TV_Past NP', 'liked': 'TV_Past NP', 'ate': 'TV_Past NP', 'saw': 'TV_Past NP', 'happily': '
Adv TV_Past NP', 'sadly': 'Adv TV_Past NP', 'nicely': 'Adv TV_Past NP'}, 'VP_P1': {'disa
ppear': 'IV_Pres_P1', 'walk': 'TV_Pres_P1', 'disappeared': 'IV_Past', 'walked': 'IV_Pas
t', 'like': 'TV_Pres_P1 NP', 'see': 'TV_Pres_P1 NP', 'shot': 'TV_Past NP', 'liked': 'TV_P
ast NP', 'ate': 'TV_Past NP', 'saw': 'TV_Past NP', 'happily': 'Adv TV_Past NP', 'sadly': '
Adv TV_Past NP', 'nicely': 'Adv TV_Past NP'}, 'N_P1': {'girls': 'girls', 'boys': 'boys',
'children': 'children', 'cars': 'cars', 'apples': 'apples', 'dogs': 'dogs'}, 'Adj': {'go
od': 'good', 'bad': 'bad', 'beautiful': 'beautiful', 'innocent': 'innocent'}, 'Adv': {'ha
ppily': 'happily', 'sadly': 'sadly', 'nicely': 'nicely'}, 'N_Sg': {'dog': 'dog', 'girl':
'girl', 'car': 'car', 'child': 'child', 'apple': 'apple', 'elephant': 'elephant'}, 'PropN
_Sg': {'vedant': 'vedant', 'ananya': 'ananya'}, 'PropN_P1': {'they': 'they', 'i': 'i'}, '
Det_Sg': {'this': 'this', 'every': 'every', 'a': 'a', 'an': 'an'}, 'Det_P1': {'these': 't
hese', 'all': 'all'}, 'Det_Both': {'some': 'some', 'the': 'the', 'several': 'several'}, '
IV_Pres_Sg': {'disappeares': 'disappeares', 'walks': 'walks'}, 'TV_Pres_Sg': {'sees': '
sees', 'likes': 'likes', 'eat': 'eat'}, 'IV_Pres_P1': {'disappear': 'disappear', 'walk'
: 'walk'}, 'TV_Pres_P1': {'see': 'see', 'like': 'like'}, 'IV_Past': {'disappeared': 'dis
appeared', 'walked': 'walked'}, 'TV_Past': {'saw': 'saw', 'liked': 'liked', 'ate': 'ate'
, 'shot': 'shot'}}

Sentence [vedant like apple] grammatically incorrect.

C:\Users\Kingsmanvk\Desktop\LP\pyParsing-master\src>
```