4. NLP – Components in NLP

Contents

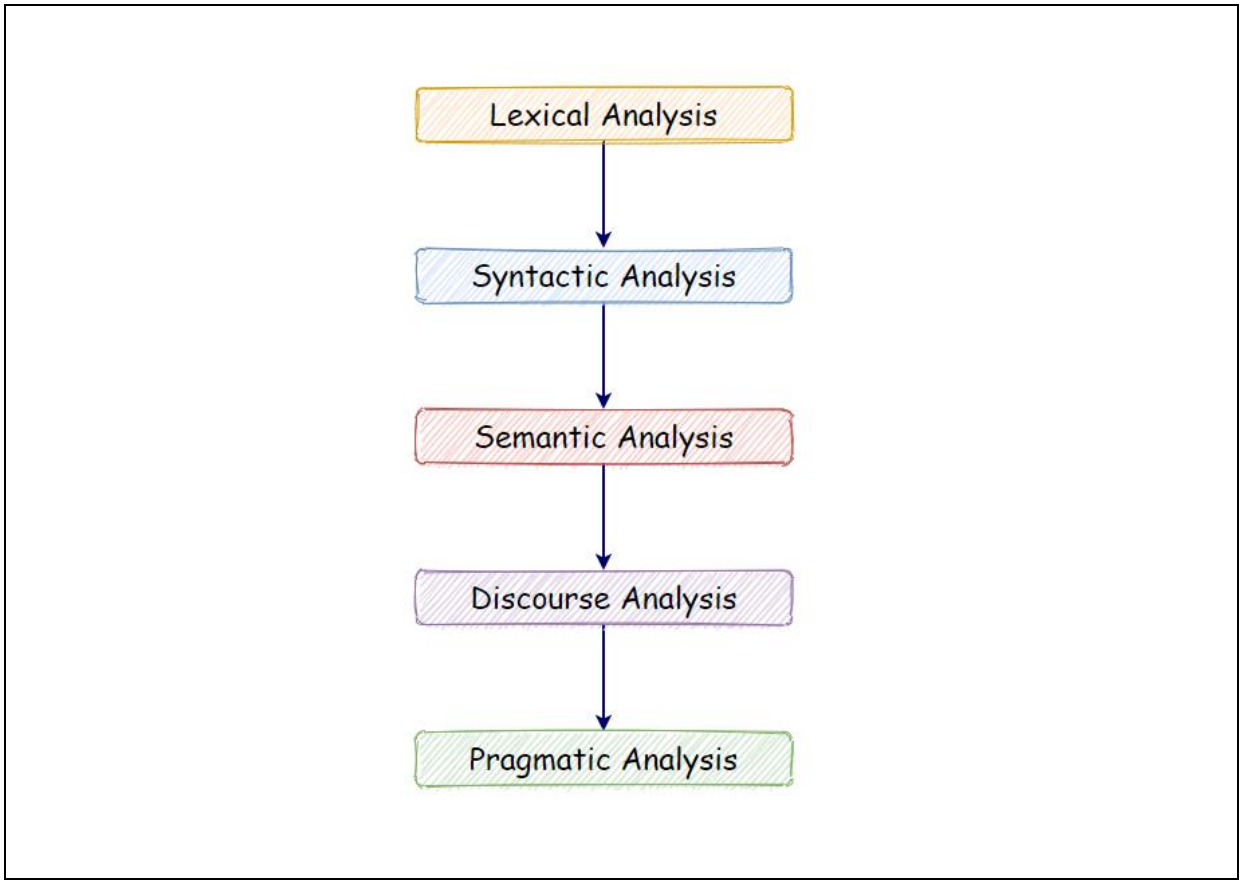
4. NLP – Components in NLP

1. Common nlp libraries

- ✓ Nltk library
- ✓ Spacy library
- ✓ TextBlob library
- ✓ Gensim framework
- ✓ Pattern framework

2. Components in NLP

- ✓ There are mainly five components in NLP
  - Lexical Analysis
  - Syntactic Analysis
  - Semantic Analysis
  - Discourse Analysis
  - Pragmatic analysis



2.1. Lexical Analysis

- ✓ Lexical means relating to the words or vocabulary of a language.
- ✓ With lexical analysis, we divide a whole chunk of text into,
  - Paragraphs.
  - Sentences.
  - Words.
- ✓ It involves identifying and analysing words' structure.

2.2. Syntactic Analysis

- ✓ Syntactic means according to the syntax
- ✓ In syntactic analysis, it is the process of analysing the words in sentence.
- ✓ Analysing the grammar and arranging the words in a manner that shows the relationship among the words.
- ✓ Example
  - The sentence "The shop goes to the house" does not pass means invalid

2.3. Semantic Analysis

- ✓ Syntactic means relating to meaning in language.
- ✓ Semantic analysis draws the exact meaning for the words, and it analyses the text to get meaningful.
- ✓ Example
  - The sentences such as "hot ice-cream" do not pass or invalid

2.4. Discourse Analysis

- ✓ Discourse means written or spoken communication.
- ✓ It considers the meaning of the sentence before it ends.
- ✓ Example
  - The sentences such as "He works at Google" in this sentence "he" should be first word in sentence

2.5. Pragmatic Analysis

- ✓ Pragmatic means practical, especially when making decisions
- ✓ Pragmatic analysis deals with overall communication and interpretation of language.
- ✓ It deals with deriving meaningful use of language in various situations.

3. Use case

- ✓ Process the below textual information

story_input.txt

Once upon a time there was an old mother pig that had three little pigs and not enough food to feed them. So, when they were old enough, she sent them out into the world to seek their fortunes.

The first little pig was very lazy. He didn't want to work at all and he built his house out of straw. The second little pig worked a little bit harder but he was somewhat lazy too and he builds his house out of sticks. Then, they sang and danced and played together and rest of the day.

The third little pig worked hard all day and built his house with bricks. It was a sturdy house complete with a fine fireplace and chimney. It looked like it could withstand the strongest winds.

Program Name: Reading textual information from file
demo1.py

```
data = open("story_input.txt")

text = data.read()

print(text)
print()
print(type(text))
print("Length of the text is:", len(text))
```

Output

```
Once upon a time there was an old mother pig that had three little pigs and not enough food to feed the
m. So, when they were old enough, she sent them out into the world to seek their fortunes.

The first little pig was very lazy. He didn't want to work at all and he built his house out of straw.
The second little pig worked a little bit harder but he was somewhat lazy too and he builds his house o
ut of sticks. Then, they sang and danced and played together and rest of the day.

The third little pig worked hard all day and built his house with bricks. It was a sturdy house complet
e with a fine fireplace and chimney. It looked like it could withstand the strongest winds.
<class 'str'>
Length of the text is: 678
```

Program Name: Sentence tokenization
demo2.py

```
import nltk
from nltk import sent_tokenize

data = open("story_input.txt")
text = data.read()

sentences = sent_tokenize(text)

print("Total sentences:", len(sentences))

for sent in sentences:
        print(sent)
```

Output

```
Total sentences: 9
Once upon a time there was an old mother pig that had three little pigs and not enough food to feed the
m.
So, when they were old enough, she sent them out into the world to seek their fortunes.
The first little pig was very lazy.
He didn't want to work at all and he built his house out of straw.
The second little pig worked a little bit harder but he was somewhat lazy too and he builds his house o
ut of sticks.
Then, they sang and danced and played together and rest of the day.
The third little pig worked hard all day and built his house with bricks.
It was a sturdy house complete with a fine fireplace and chimney.
It looked like it could withstand the strongest winds.
```

Program Name: Word tokenization
demo3.py

```
import nltk
from nltk import word_tokenize

data = open("story_input.txt")
text = data.read()

words = word_tokenize(text)
print(words)
```

Output

```
['Once', 'upon', 'a', 'time', 'there', 'was', 'an', 'old', 'mother', 'pig', 'that', 'had', 'three', 'li
ttle', 'pigs', 'and', 'not', 'enough', 'food', 'to', 'feed', 'them', '.', 'So', ',', 'when', 'they', 'w
ere', 'old', 'enough', ',', 'she', 'sent', 'them', 'out', 'into', 'the', 'world', 'to', 'seek', 'their'
, 'fortunes', '.', 'The', 'first', 'little', 'pig', 'was', 'very', 'lazy', '.', 'He', 'did', "n't", 'wa
nt', 'to', 'work', 'at', 'all', 'and', 'he', 'built', 'his', 'house', 'out', 'of', 'straw', '.', 'The',
 'second', 'little', 'pig', 'worked', 'a', 'little', 'bit', 'harder', 'but', 'he', 'was', 'somewhat', '
lazy', 'too', 'and', 'he', 'builds', 'his', 'house', 'out', 'of', 'sticks', '.', 'Then', ',', 'they', '
sang', 'and', 'danced', 'and', 'played', 'together', 'and', 'rest', 'of', 'the', 'day', '.', 'The', 'th
ird', 'little', 'pig', 'worked', 'hard', 'all', 'day', 'and', 'built', 'his', 'house', 'with', 'bricks'
, '.', 'It', 'was', 'a', 'sturdy', 'house', 'complete', 'with', 'a', 'fine', 'fireplace', 'and', 'chimn
ey', '.', 'It', 'looked', 'like', 'it', 'could', 'withstand', 'the', 'strongest', 'winds', '.']
```

Program Name: Finding word frequency
demo4.py

```
import nltk
from nltk import word_tokenize
from nltk.probability import FreqDist

data = open("story_input.txt")
text = data.read()

words = word_tokenize(text)

fdist = FreqDist(words)
result = fdist.most_common(10)
print(result)
```

Output

```
[('.', 9),
('and', 8),
('little', 5),
('a', 4),
('was', 4),
('pig', 4),
('house', 4),
('to', 3),
(',', 3),
('out', 3)]
```

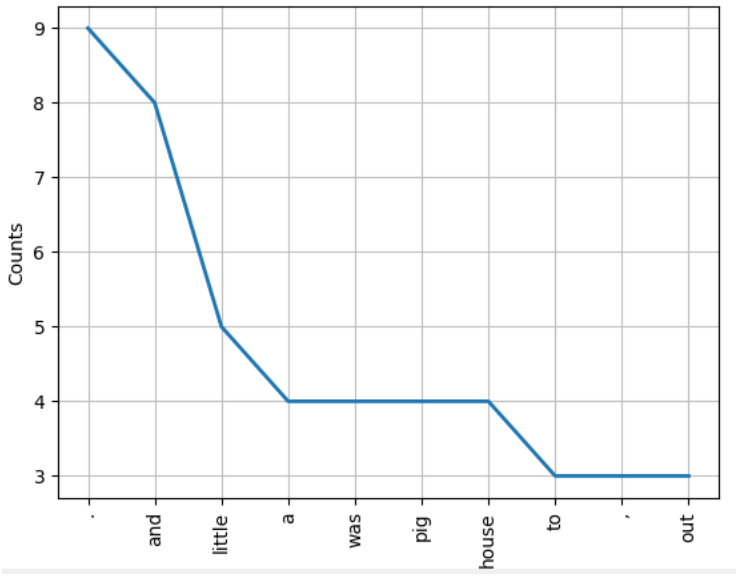
Note

- ✓ Notice that the most used words are punctuation marks and stopwords.
- ✓ We will have to remove such words to analyse the actual text.

Program Name: Plotting the frequency graph of words with punctuations
demo5.py

```
import nltk
from nltk import word_tokenize
from nltk.probability import FreqDist
import matplotlib.pyplot as plt

data = open("story_input.txt")
text = data.read()

words = word_tokenize(text)

fdist = FreqDist(words)
fdist.plot(10)
```

Output



- ✓ In the graph above, notice that a period "." is used nine times in our text.
- ✓ Analytically speaking, punctuation marks are not that important for natural language processing.
- ✓ Therefore, in the next step, we will be removing such punctuation marks.

Program Name: Removing the punctuation marks
demo6.py

```
import nltk
from nltk import word_tokenize

data = open("story_input.txt")
text = data.read()

words = word_tokenize(text)

words_no_punc = []

for w in words:
        if w.isalpha():
                words_no_punc.append(w.lower())

print(words_no_punc)
print("Length of words:", len(words_no_punc))
```

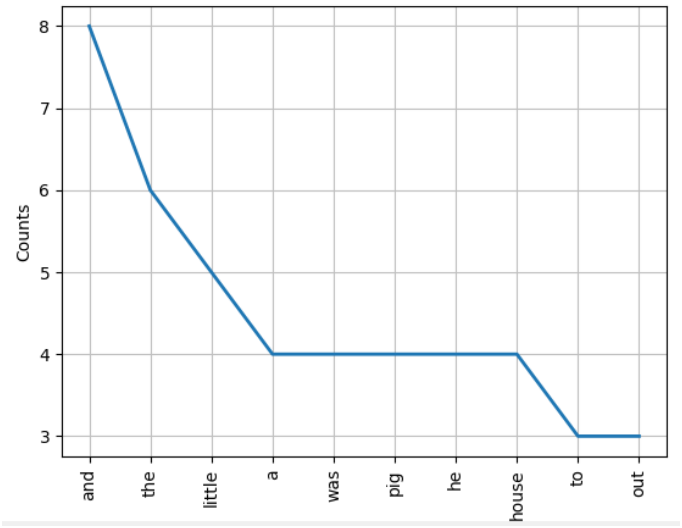
Output

```
['once', 'upon', 'a', 'time', 'there', 'was', 'an', 'old', 'mother', 'pig', 'that', 'had', 'three', 'li
ttle', 'pigs', 'and', 'not', 'enough', 'food', 'to', 'feed', 'them', 'so', 'when', 'they', 'were', 'old
', 'enough', 'she', 'sent', 'them', 'out', 'into', 'the', 'world', 'to', 'seek', 'their', 'fortunes', '
the', 'first', 'little', 'pig', 'was', 'very', 'lazy', 'he', 'did', 'want', 'to', 'work', 'at', 'all',
'and', 'he', 'built', 'his', 'house', 'out', 'of', 'straw', 'the', 'second', 'little', 'pig', 'worked',
 'a', 'little', 'bit', 'harder', 'but', 'he', 'was', 'somewhat', 'lazy', 'too', 'and', 'he', 'builds',
'his', 'house', 'out', 'of', 'sticks', 'then', 'they', 'sang', 'and', 'danced', 'and', 'played', 'toget
her', 'and', 'rest', 'of', 'the', 'day', 'the', 'third', 'little', 'pig', 'worked', 'hard', 'all', 'day
', 'and', 'built', 'his', 'house', 'with', 'bricks', 'it', 'was', 'a', 'sturdy', 'house', 'complete', '
with', 'a', 'fine', 'fireplace', 'and', 'chimney', 'it', 'looked', 'like', 'it', 'could', 'withstand',
'the', 'strongest', 'winds']
Lenght of words: 132
```

Program Name: Plotting the frequency graph of words without punctuations
demo7.py

```
import nltk
from nltk import word_tokenize
from nltk.probability import FreqDist
import matplotlib.pyplot as plt

data = open("story_input.txt")
text = data.read()

words = word_tokenize(text)

words_no_punc = []

for w in words:
        if w.isalpha():
                words_no_punc.append(w.lower())

fdist = FreqDist(words_no_punc)
fdist.plot(10)
```

Output



- ✓ Notice that we still have many words that are not very useful in the analysis of our text file sample, such as “and,” “but,” “so,” and others.
- ✓ Next, we need to remove coordinating conjunctions.

Program Name: List of the stop words
demo8.py

```
from nltk.corpus import stopwords

stopwords = stopwords.words('english')
print(stopwords)
```

Output

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you
'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', '
hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what
', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were
', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the
', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about
', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from'
, 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here',
'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', '
some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can',
 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'a
in', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'h
asn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'ne
edn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won',
 "won't", 'wouldn', "wouldn't"]
```

Program Name: Removing the stop words
demo9.py

```
import nltk
from nltk import word_tokenize
from nltk.probability import FreqDist
import matplotlib.pyplot as plt
from nltk.corpus import stopwords

data = open("story_input.txt")
text = data.read()

stopwords = stopwords.words('english')
words = word_tokenize(text)


words_no_punc = []

for w in words:
    if w.isalpha():
        words_no_punc.append(w.lower())


clean_words = []

for w in words_no_punc:
    if w not in stopwords:
        clean_words.append(w)

print(clean_words)
print("Length of clean words:", len(clean_words))
```

Output

```
['upon', 'time', 'old', 'mother', 'pig', 'three', 'little', 'pigs', 'enough', 'food', 'feed', 'old', 'e
nough', 'sent', 'world', 'seek', 'fortunes', 'first', 'little', 'pig', 'lazy', 'want', 'work', 'built',
 'house', 'straw', 'second', 'little', 'pig', 'worked', 'little', 'bit', 'harder', 'somewhat', 'lazy',
'builds', 'house', 'sticks', 'sang', 'danced', 'played', 'together', 'rest', 'day', 'third', 'little',
'pig', 'worked', 'hard', 'day', 'built', 'house', 'bricks', 'sturdy', 'house', 'complete', 'fine', 'fir
eplace', 'chimney', 'looked', 'like', 'could', 'withstand', 'strongest', 'winds']
Lenght of clean words 65
```

Program Name: Word frequency distribution
demo10.py

```
import nltk
from nltk import word_tokenize
from nltk.probability import FreqDist
import matplotlib.pyplot as plt
from nltk.corpus import stopwords

data = open("story_input.txt")
text = data.read()

stopwords = stopwords.words('english')
words = word_tokenize(text)

words_no_punc = []

for w in words:
    if w.isalpha():
        words_no_punc.append(w.lower())

clean_words = []

for w in words_no_punc:
    if w not in stopwords:
        clean_words.append(w)

fdist = FreqDist(clean_words)
result = fdist.most_common(10)
print(result)
```

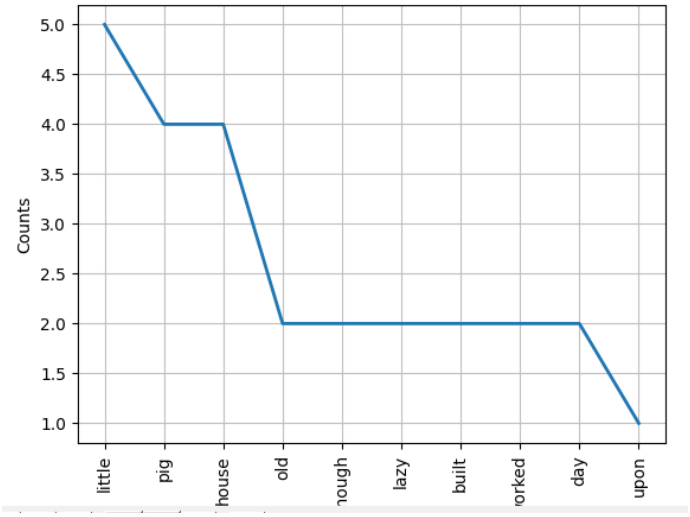
Output

```
[('little', 5),
('pig', 4),
('house', 4),
('old', 2),
('enough', 2),
('lazy', 2),
('built', 2),
('worked', 2),
('day', 2),
('upon', 1)]
```

Program Name: Plotting the useful words
demo11.py

```
import nltk
from nltk import word_tokenize
from nltk.probability import FreqDist
import matplotlib.pyplot as plt
from nltk.corpus import stopwords

data = open("story_input.txt")
text = data.read()

stopwords = stopwords.words('english')
words = word_tokenize(text)

words_no_punc = []

for w in words:
        if w.isalpha():
                words_no_punc.append(w.lower())


clean_words = []

for w in words_no_punc:
        if w not in stopwords:
                clean_words.append(w)


fdist = FreqDist(clean_words)
fdist.most_common(10)
fdist.plot(10)
```

Output



✓ As shown above, the final graph has many useful words that help us understand what our sample data is about, showing how essential it is to perform data cleaning on NLP.

4. Word Cloud

- ✓ Word Cloud is a data visualization technique.
- ✓ In which words from a given text display on the main chart.
- ✓ In this technique, more frequent or essential words display in a larger and bolder font.
- ✓ Less frequent or essential words display in smaller or thinner fonts.
- ✓ It is a beneficial technique in NLP that gives us a glance at what text should be analysed.

Program Name: Wordcloud example
demo12.py

```
import matplotlib.pyplot as plt
from wordcloud import WordCloud

text = "Python is good programming language, Python is very easy, Learning Data Science starts from Python"

wordcloud = WordCloud().generate(text)

plt.figure(figsize = (12, 12))
plt.imshow(wordcloud)

plt.axis('off')
plt.show()
```

Output