FSC CS Department Workshop Presentation

NLP

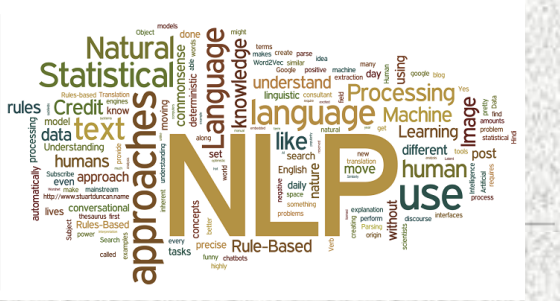
Compiled by Grace Zhao

10/28/21

Lecture Outline

- Intro to NLP
  - Intro to Natural Language Generation
  - Intro to Natural Language Understanding

What is NLP



- Natural Language Processing
- NLP is a branch of Artificial Intelligence (AI) that enables machines to understand the human language, both written and spoken.
- A combination of statistics, computational linguistics, and computer science
- NLP $\subset$ Deep Learning $\subset$ ML $\subset$ AI

How Does It Work?

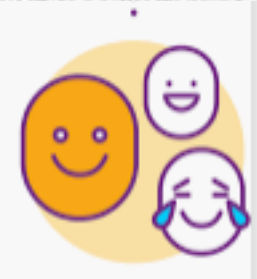
- Behind the scenes, NLP analyzes the grammatical structure of sentences and the individual meaning of words, then uses algorithms to extract meaning and deliver outputs.

NLP in Everyday Apps

- Speech recognition
  - Probably, the most popular examples of NLP in action are virtual assistants, like Google Assist, Siri, and Alexa. NLP understands written and spoken text like *"Hey Siri, where is the nearest gas station?"* and transforms it into numbers, making it easy for machines to understand
- Chatbots.
  - They help support teams solve issues by understanding common language requests and responding automatically.
- Spell checking

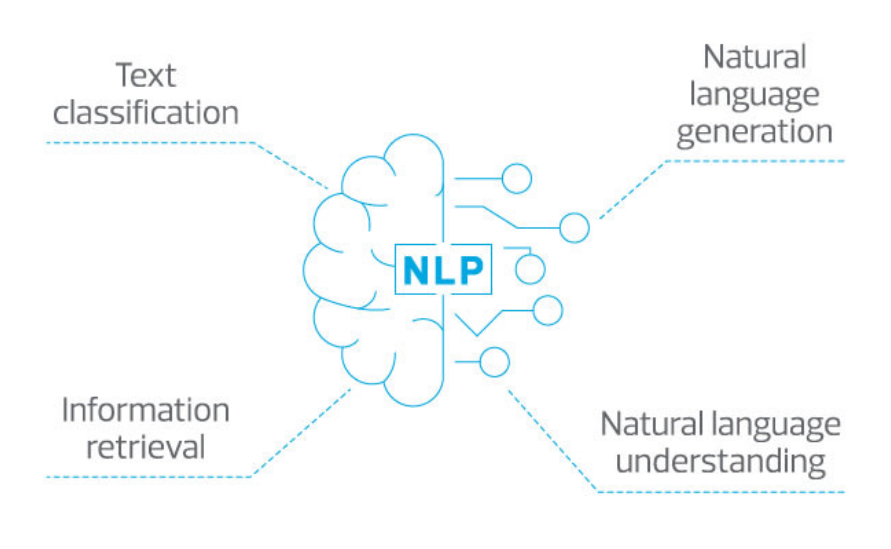
NLP in Everyday Apps

- Text Search
- Language translation
- Advertisement match
  - Based on your search text data and match it with the text data of advertisement
- Sentimental analysis



Text Mining and Information Retrieval

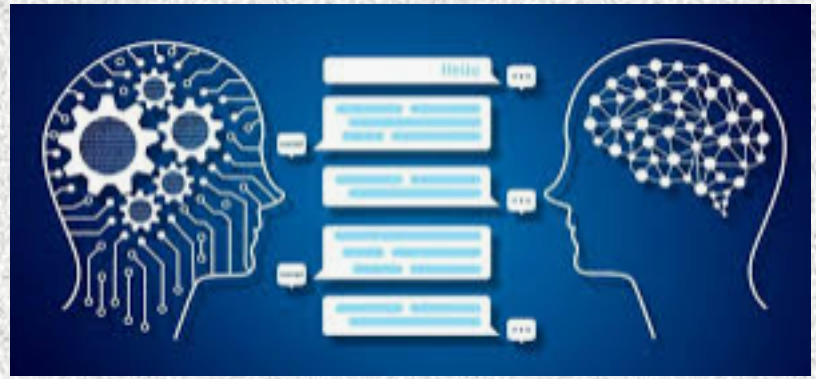
- Text Mining Process of deriving meaningful information from natural language.
- Information retrieval may be defined as a software program that deals with the organization, storage, retrieval and evaluation of information from document repositories particularly textual information.



https://www.tutorialspoint.com/natural_language_processing/natural_language_processing_information_retrieval.htm

Components of NLP

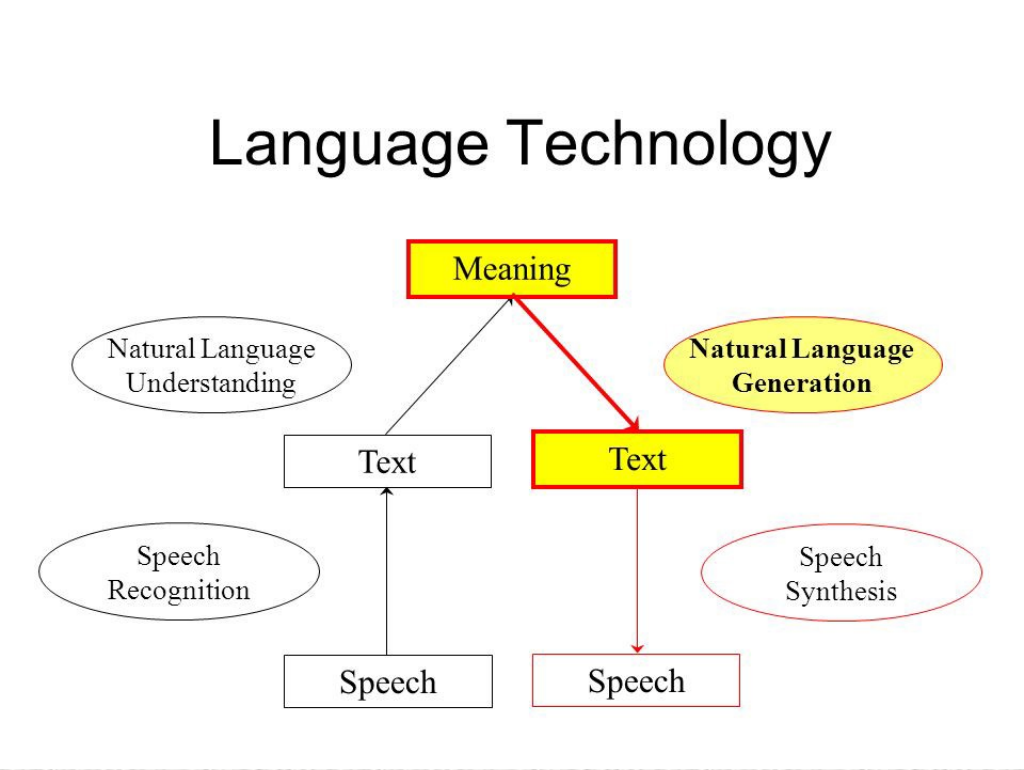
- Natural language understanding (NLU)
  - Mapping language into useful representations
  - Analyzing different aspects of the language
- Natural language generation (NLG)
  - Process of producing some meaningful sentences and phrases in the form of natural language from some internal representation.



Which one is more difficult, do you think?

Natural Language Generation

- Text Planning
  - Retrieving the relevant contents from the knowledge base
- Sentence Planning
  - Choosing meaningful words from meaningful phrases
  - Setting tone of the sentences
- Text Realization
  - Mapping sentence plan to sentence structure



NLG Process

- You train a Natural Language Generation software to pull out **relevant information** from a set of **structured data** and include the information in a sentence or paragraph.

Applications of NLG

- Automate readable financial or medical reports from data
  - One of the primary design functionalities of NLG is the ability to turn raw information into easy-to-understand reports and documents, especially when it comes to high volumes of data or information that require precise translation.
  - Virtual Assistant, Chatbot, AI customer service.
  - Analytics Reporting
  - Content automation

https://www.analyticssteps.com/blogs/natural-language-generation-nlg-types-working-and-applications

Six Stages of NLG

- **Examine the content:**

  Data is filtered to identify what should be included in the final content. Identifying the key subjects in the source document, as well as the relationships between them, is part of this stage.

- **Data comprehension:**

  The data is analyzed, patterns are discovered, and the information is placed in context. At this point, machine learning is frequently applied.

- **Document structuring:**

  Based on the sort of data being analyzed, a documented plan is constructed and a narrative framework is chosen.

- **Sentence aggregation:**

  It is a technique for combining sentences. Sentences or sections of sentences that are relevant to the issue are mixed in ways that accurately summarize the topic.

- **Grammatical organization:**

  To generate natural-sounding writing, grammatical rules are employed. The sentence's syntactical structure is deduced by the software. It then rewrites the statement in a grammatically accurate manner using this information.

- **Presentation of the language:**

  The final output is generated using a template or format chosen by the user or programmer.

Natural Language Understanding

Ambiguity of Natural Languages

- Human language
  - `alphabets->words-sentences->languages (`*`rules/grammars`*`)`
  - Only 21% data is presented in a structured form. Emails, social media contain highly unstructured data.
- Ambiguity
  - Lexical (semantic)
    - Two or more possible meanings within a single word
  - Syntactical (grammatical, structural)
    - Presence of two or more meanings within a single sentence or a sequence of words
    - Ex: The chicken is ready to eat. I saw the man with the binoculars.
  - Referential
    - Ambiguity arises when we refer to something using pronouns.
    - Ex: The boy told his father the theft. *He* was very upset.

Various Steps Involved in NLP

- Tokenization
- POS Tag
  - Part-of-speech
- Name entity recognition
- Chunking
  - Pick up individual information and piece together

NLTK

- **`nltk`** *`(natural language toolkits)`* is a Python library for processing natural languages enabling easy access to "text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum".
- `%pip install nltk`
- `>>> nltk.download()`

```
>>> import nltk
>>> from nltk.corpus import brown
>>> brown.words()
>>> nltk.corpus.gutenberg.fileids()
```

Tokenization

- Tokenization
  - First step in NLP
  - Breaking a string into smaller tokens/units that can be used for tokenization
- 3-Step Process
  - Break a complex sentence into words
  - Understand the importance of each of the words with respect to the sentence
  - Produce a structural description on an input sentence
    - Ex: Today we will understand tokenization .

```
>>> AI = "Today  we will understand tokenization."
>>> from nltk.tokenize import word_tokenize
>>> tprint(word_tokenize(AI))
['Today', 'we', 'will', 'understand', 'tokenization', '.']
```

Tokenization

- Ngram
- Trigram
- Bigram

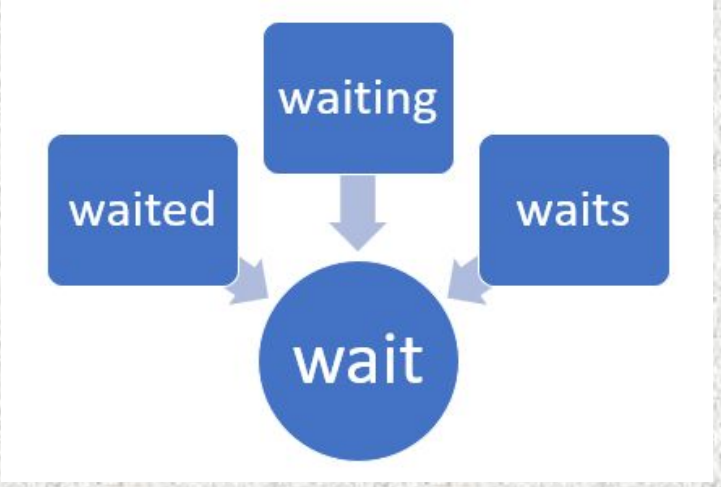

```
>>>from nltk.tokenize import word_tokenize
>>> from nltk.util import bigrams, trigrams, ngrams
```

```
>>> tokens = nltk.word_tokenize(sent)

>>> bigram = list(nltk.bigrams(tokens))
>>> trigram = list(nltk.trigrams(tokens))
>>> ngram = list(nltk.ngrams(tokens,5))
```

Stemming

- Now let's make some changes to the tokens. For that we have "stemming."
- Stemming
  - Normalize words into its base form or root form
  - Ex: Affectation, affects, affections, affected, affection, affecting - *Affect*
- Stemming algorithm
  - Works by cutting off the beginning or the end of the word, considering a list of common prefixes or suffixes that can be found in an infected word.
  - Can be successful on some occasions, but not always.
- Inflection is the modification of a word to express different grammatical categories such as tense, case, voice, aspect, person, number, gender, and mood.

Limitation



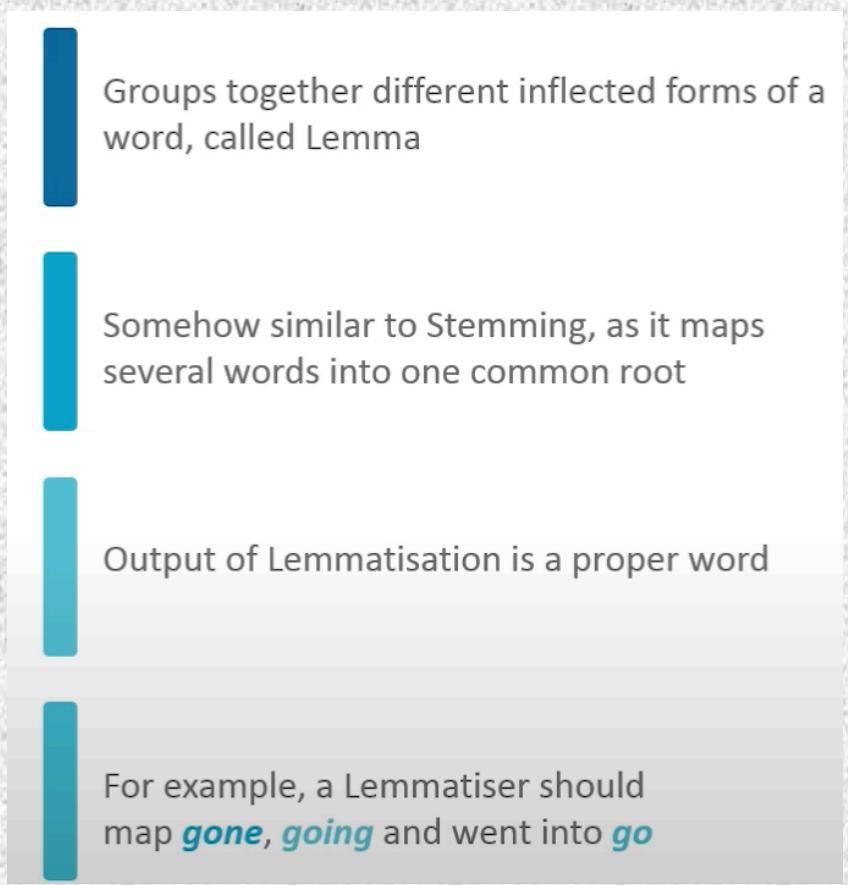
Stemmer

- Porter's Stemmer (created in 1989)
  - It first mention was in 1980 in the paper *An algorithm for suffix stripping* by Martin Porter and it is one of the widely used stemmers available in ***nltk***.
- Snowball Stemmer (Porter 2)
  - An improved version from Porter's Stemmer.
- Lancaster Stemmer
  - Most aggressive one

```
>>> from nltk.stem import PorterStemmer,
SnowballStemmer, LancasterStemmer
>>> port= PorterStemmer()
>>> port.stem("beautiful")

>>> snow= SnowballStemmer("english")
>>> snow.stem("beautiful")

>>> lan= lancasterStemmer()
>>> lan.stem("beautiful")

>>> words =
["beautiful","beauty","beauties","beautifully",
"salty", "generous", "corpora"]
>>> list(map(port.stem,words))
>>> list(map(snow.stem,words))
>>> list(map(lan.stem,words))
```

Lemmatization

- Lemmatization
  - Morphological analysis of the word
    - Morphological analysis or general morphological analysis is a method for exploring possible solutions to a multi-dimensional, non-quantified complex problem. (wiki)
  - Group together different infected forms of the word
  - *Root word* known as "lemma"
  - If not assign any POS Tags, so it assumes every word is a noun.



Using the WordNet lemmatizer

- WordNet is a lexical database of the English language. In this database nouns, verbs, adjectives, and adverbs are grouped together as sets. This helps in the lemmatization process.

```
>>> from nltk.stem import wordnet
>>> from nltk.stem import WordNetLemmatizer
>>> lem=WordNetLemmatizer()
>>> lem.lemmatize("corpora")


>>> list(map(lem.lemmatize,tokens))
```

Difference between Stemming and lemmatization

S.No	Stemming	Lemmatization
1	Stemming is faster because it chops words without knowing the context of the word in given sentences.	Lemmatization is slower as compared to stemming but it knows the context of the word before proceeding.
2	It is a rule-based approach.	It is a dictionary-based approach.
3	Accuracy is less.	Accuracy is more as compared to Stemming.
4	When we convert any word into root-form then stemming may create the non-existence meaning of a word.	Lemmatization always gives the dictionary meaning word while converting into root-form.
5	Stemming is preferred when the meaning of the word is not important for analysis. Example: Spam Detection	Lemmatization would be recommended when the meaning of the word is important for analysis. Example: Question Answer
6	For Example: “Studies” => “Studi”	For Example: “Studies” => “Study”

Stop Words

- Helpful in formation of the sentences, but not necessarily helpful in language processing.

```
>>> import nltk
>>> nltk.download('stopwords')

>>>from nltk.corpus import stopwords
>>> stopwords.words('english')
```


```
>>> import re
>>> punctuation=re.compile(r'[-.?!,:;()|0-9]')

>>> sent = "Oy, you, what's up?"
>>> tokens = nltk.word_tokenize(sent)
>>> stems = ' '.join(list(map(lambda x:punctuation.sub('',x), tokens)))
```

Filter Stop Words and Punctuations

```
import nltk
from nltk.corpus import stopwords

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

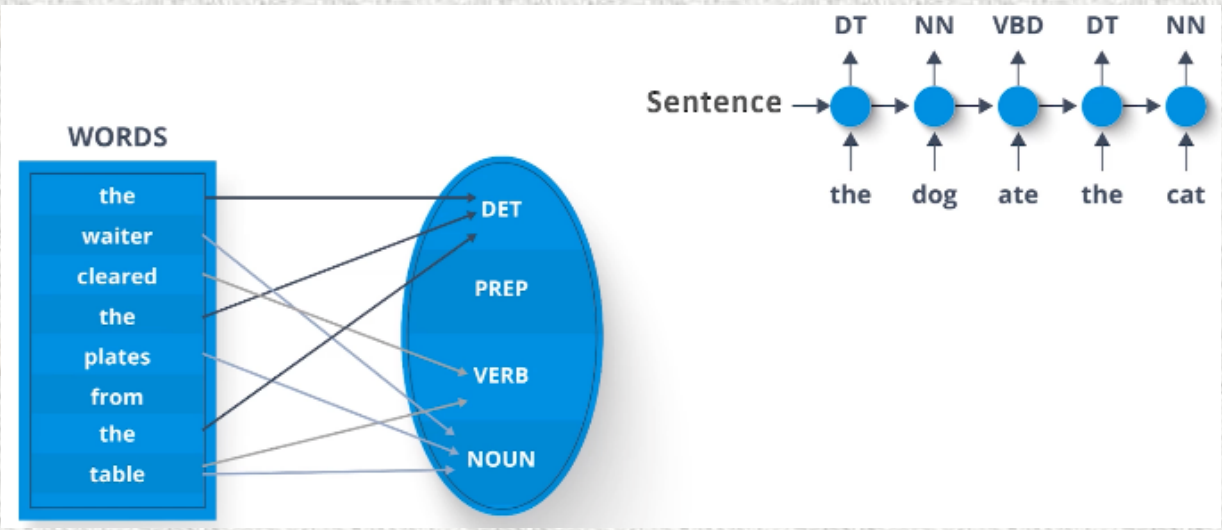
```
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords
import  re
punctuation=re.compile(r'[-.?!,:;()|0-9]')

data = "All work, no play makes Jacky dull boy. All play and no work makes Joey a dull boy. You've got to be joking!"
stops = set(stopwords.words('english'))
tokens = word_tokenize(data)

newtokens = list(filter(lambda x: (x not in stops), tokens))
stems = ' '.join(list(map(lambda x:punctuation.sub('',x), newtokens)))
```

Parts of Speech (POS)

- A word can be a noun or a verb.

```
>>> sent="The dog ate the cat."
>>> pos_tokens=word_tokenize(sent)
>>> pos= nltk.pos_tag(pos_tokens)
>>> pos
```

[('The', 'DT'), ('dog', 'NN'), ('ate', 'VBD'), ('the', 'DT'), ('cat', 'NN'), ('.', '.')]



POS Tags

Tag	Description
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun

Tag	Description
PRP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBR	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Whdeterminer
WP	Whpronoun
WP$	Possessive whpronoun
WRB	Whadverb

Name Entity Recognition (NER)

- Also called **entity identification** or **entity extraction** – is a NLP technique that automatically identifies named entities in a text and classifies them into predefined categories.
- Google Knowledge Graph, Wikipedia, IBM Watson
- Three phases
  - Noun phrase recognition
    - extracting all the noun phases from a text, using
      - dependency passing
      - POS tagging
  - Phrase recognition
  - Entity disambiguation
    - Sometime entities are misclassified

Example

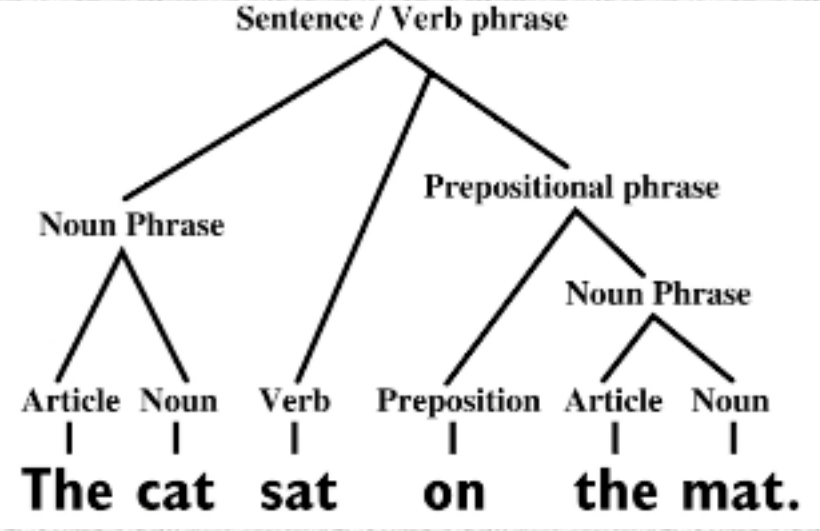
```
>>> st="The CS department is in the Whitman Hall."
>>>chunks=ne_chunk(nltk.pos_tag(word_tokenize(st))) #ne: named entity
>>> chunks
```

```
Tree('S', [('The', 'DT'), Tree('ORGANIZATION', [('CS', 'NNP')]), ('department', 'NN'), ('is',
'VBZ'), ('in', 'IN'), ('the', 'DT'), Tree('FACILITY', [('Whitman', 'NNP'), ('Hall', 'NNP')]), ('.',
'.')])
```

```
>>> print(chunks)
(S
  The/DT
  (ORGANIZATION CS/NNP)
  department/NN
  is/VBZ
  in/IN
  the/DT
  (FACILITY Whitman/NNP Hall/NNP)
  ./.)
```

Syntax Tree

- A tree representation of syntactic structure of sentences or strings
- A syntax tree is composed of basic constituents to construct a sentence and languages, including words and rules.
- GhostScript



Chunking

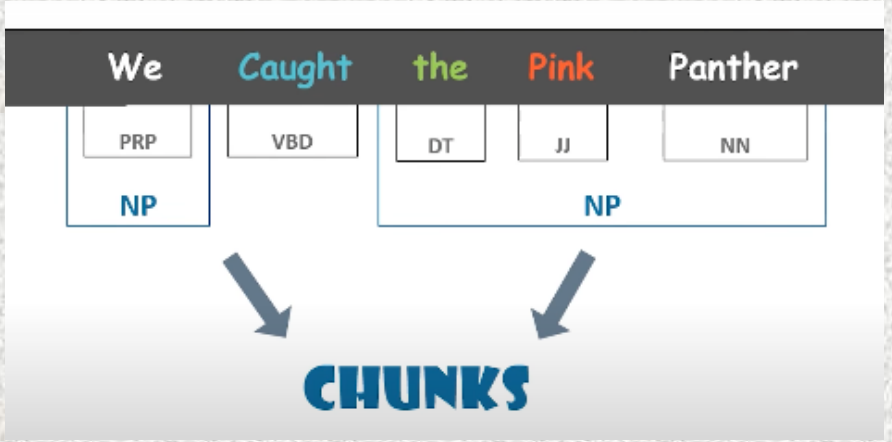
- Picking up *individual* pieces of information and *grouping* them into bigger pieces.
- Process of extracting a group of words or phrases from an unstructured text..

```
st="We caught the pink panther."
tokens=nltk.pos_tag(word_tokenize(st))
grammer= r"NP:{<DT>?(<JJ>|<NN>)*(<NN>|<PRP>)}"
parser=nltk.RegexpParser(grammer)
result=parser.parse(tokens)
print(result)
```

```
(S
(NP We/PRP)
caught/VBD
(NP the/DT pink/NN panther/NN)
./.
)
```

```
parser.parse(tokens).draw()
#graphical tree display
```

```
from nltk import RegexpParser
```



Some NLP Basic Terms

- Vocabulary
  - The entire set of terms used in a body of text.
- Documents
  - Document refers to a body of text. A collection of documents make up a corpus. For instance, a movie review.
- Corpus (pl. corpora)
  - Body of text, singular. Corpora is the plural of this. Example: A collection of medical journals.
- Lexicon
  - Words and their meanings. Example: English dictionary. There is a special lexicon for financial investors, doctors, children, mechanics, and so on.
- Token
  - Each "entity" that is a part of whatever was split up based on rules. For examples, each word is a token when a sentence is "tokenized" into words. Each sentence can also be a token, if you tokenized the sentences out of a paragraph.

Some NLP Basic Terms

- (Word) Embeddings
  - Each token is embedded as a vector before it can be passed to a machine learning model. While generally referred to as word embeddings, embeddings can be created on the character or phrase level as well.
- n-grams
  - A contiguous sequence of *n* tokens in a given text. In the phrase *the day is young,* we have bi-grams (the, day), (day, is), (is, young).
- Parts of Speech (POS)
  - The syntactic function of a word. We are all probably familiar with the different parts of speech in English: noun, verb, adjective, adverb … etc.
- Normalization
  - The process of reducing similar tokens to a canonical form. For instance, if we believe *hello* and *Hello* are for all intents and purposes the same, we can *normalize* our text by mapping both terms to *hello.*
- Transformers
  - A new deep learning architecture introduced in 2017 which surpassed several prior benchmarks for NLP tasks. Transformers compensate for two of the Recurrent Neural Network's shortcomings: ability to parallelize computations and is better suited to learn long-term dependencies between words.

References

- https://www.edureka.co/python-natural-language-processing-course
- https://sdsclub.com/bert-google-nlp-algorithm/
- https://towardsml.com/2019/09/17/bert-explained-a-complete-guide-with-theory-and-tutorial/
- https://blog.finxter.com/how-to-install-nltk-on-pycharm/
- https://monkeylearn.com/blog/nlp-ai/
- https://machinelearningknowledge.ai/beginners-guide-to-stemming-in-python-nltk/
- https://medium.com/analytics-vidhya/nlp-glossary-for-beginners-c3093529ee4
- https://www.analyticssteps.com/blogs/what-stemming-and-lemmatization-nlp
- https://pythonwife.com/lemmatization-in-nlp/
- https://pythonspot.com/nltk-stop-words/