19CSE453 – Natural Language Processing

Human Languages, Models

By

Ms. Kavitha C.R.

Dept. of Computer Science and Engineering

Amrita School of Engineering, Bengaluru

Amrita Vishwa Vidyapeetham

Human Languages

Human language is distinct from all other known animal forms of communication in being compositional.

Human language allows speakers to express thoughts in sentences comprising subjects, verbs and objects such as 'I kicked the ball' and recognizing past, present and future tenses.

A natural language is **a human language**, such as English language,

An artificial language, a machine language, or the language of formal logic.
Also called ordinary language.

The theory of universal grammar proposes that all-natural languages have certain underlying rules that shape and limit the structure of the specific grammar for any given language.

Natural language processing (also known as *computational linguistics*) is the scientific study of language from a computational perspective, with a focus on the interactions between natural (human) languages and computers.

Observations

The term 'natural language' is used in opposition to the terms 'formal language' and 'artificial language,' but the important difference is that natural languages are not *actually constructed* as artificial languages and they do not *actually appear* as formal languages.

But they are considered and studied as though they were formal languages 'in principle'.

Behind the complex surface of natural languages there are
— according to this way of thinking
— rules and principles that determine their constitution and functions

Essential Concepts

• All languages are systematic. They are governed by a set of interrelated systems that include phonology(sounds and sign lang),graphics(usually), morphology, syntax, lexicon, and semantics.
• All natural languages are conventional and arbitrary. They obey rules, such as assigning a particular word to a particular thing or concept. But there is no reason that this particular word was originally assigned to this particular thing or concept.
• All natural languages are redundant, meaning that the information in a sentence is signaled in more than one way.
• All natural languages change. There are various ways a language can change and various reasons for this change.

There are around 7,000 human languages worldwide, and while they're all unique, they're also more similar
– particularly when it comes to the grammar, or the way that sentences can be formed and used.

Types of Human Languages

What are the types of human language?

Three means of human communication:

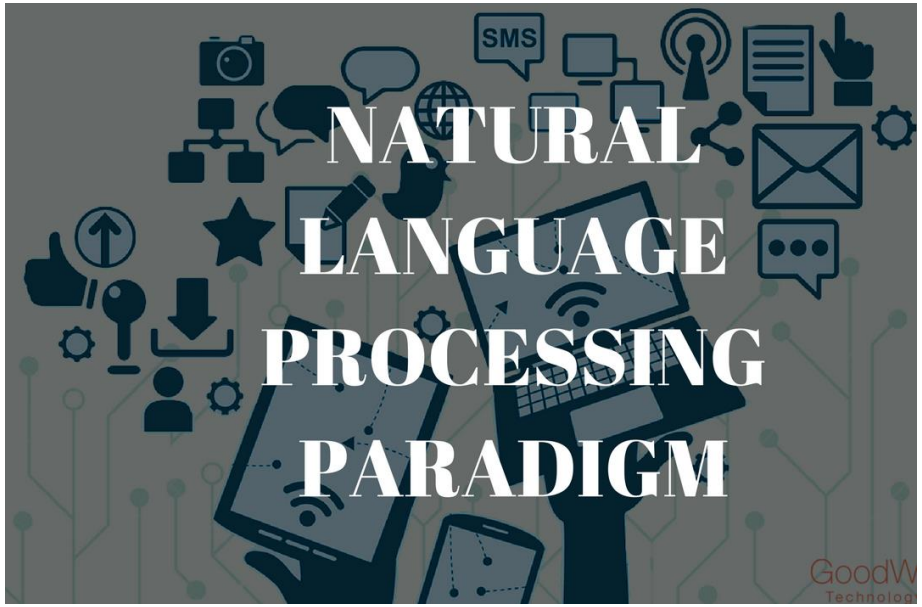
speech,
writing, and
gesture

Of particular interest will be how each form relates to language itself and the required features for communication to count as a language.

NLP Paradigm

1. Knowledge-based methods
– Rely on the manual encoding of linguistic knowledge
» E.g. Finite State Automata(FSA) for morphological parsing(structure), pronunciation variation rules

2. Statistical / learning methods
– Rely on the automatic acquisition of linguistic knowledge from corpora
– Data-driven, corpus-based methods



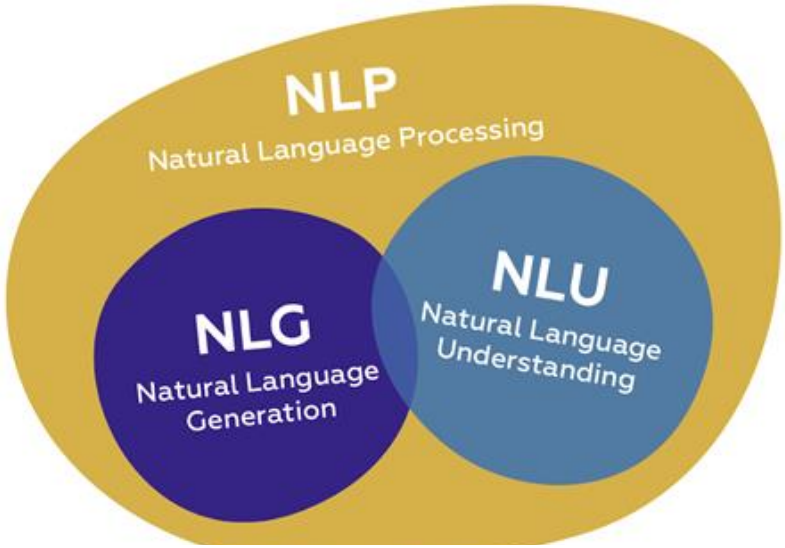
Corpora is a large collection of text electronically stored on computers which contains authentic language.

Components of NLP

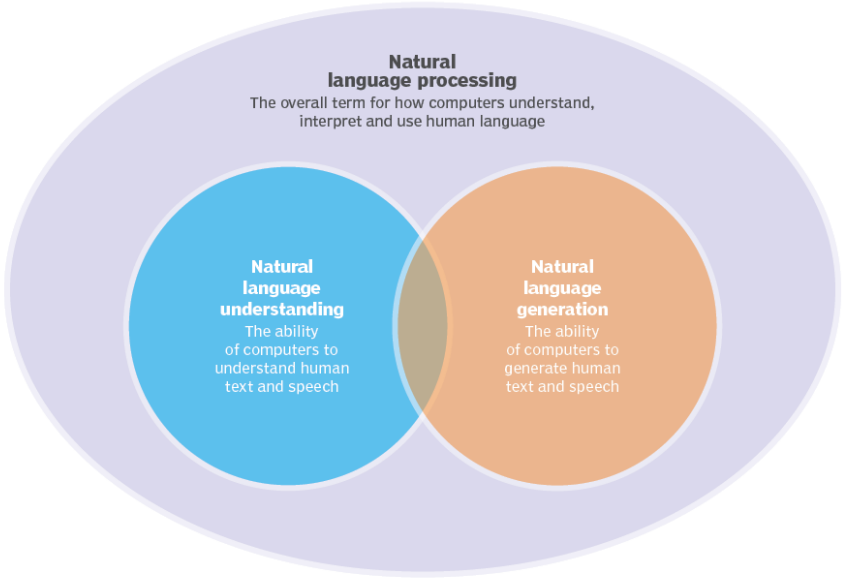
Natural Language Processing (NLP)

1. Natural Language Understanding
   - Taking some spoken/typed sentence and working out what it means
2. Natural Language Generation
   - Taking some formal representation of what you want to say and working out a way to express it in a natural (human) language (e.g., English)

NLP lies at the intersection of computational linguistics and machine learning.

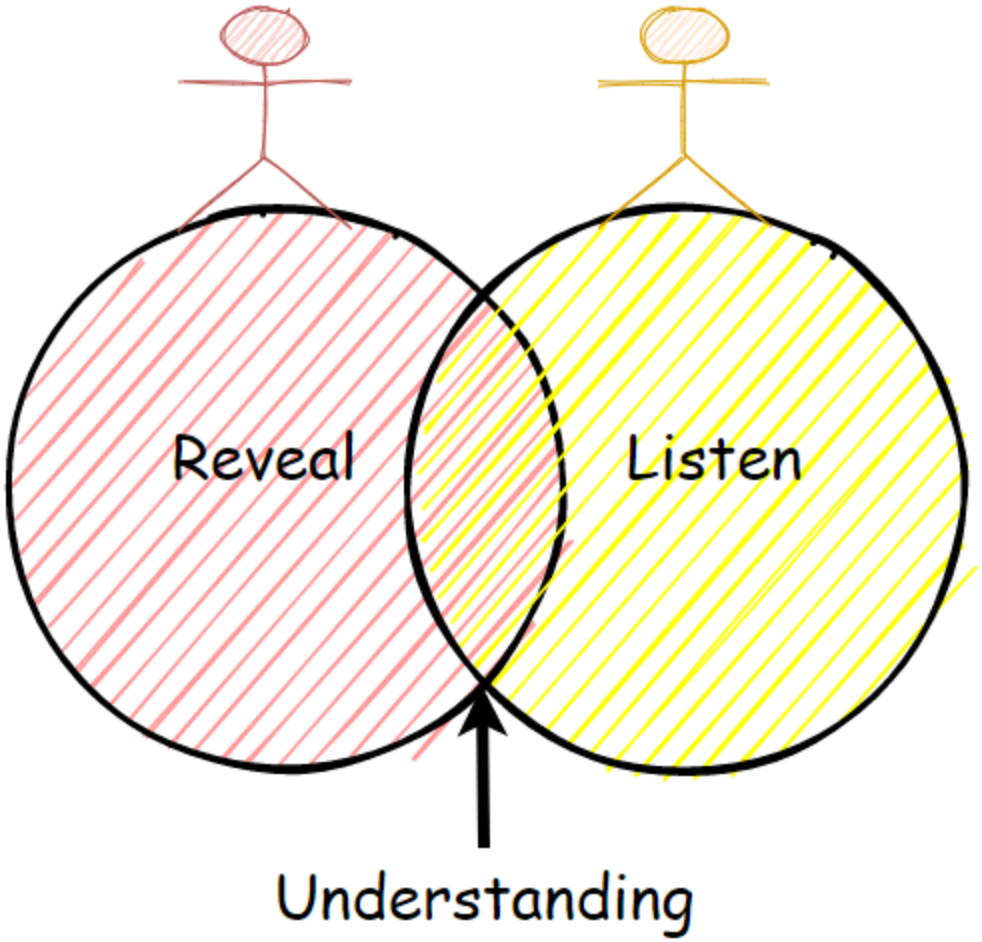


How NLP, NLU and NLG are related

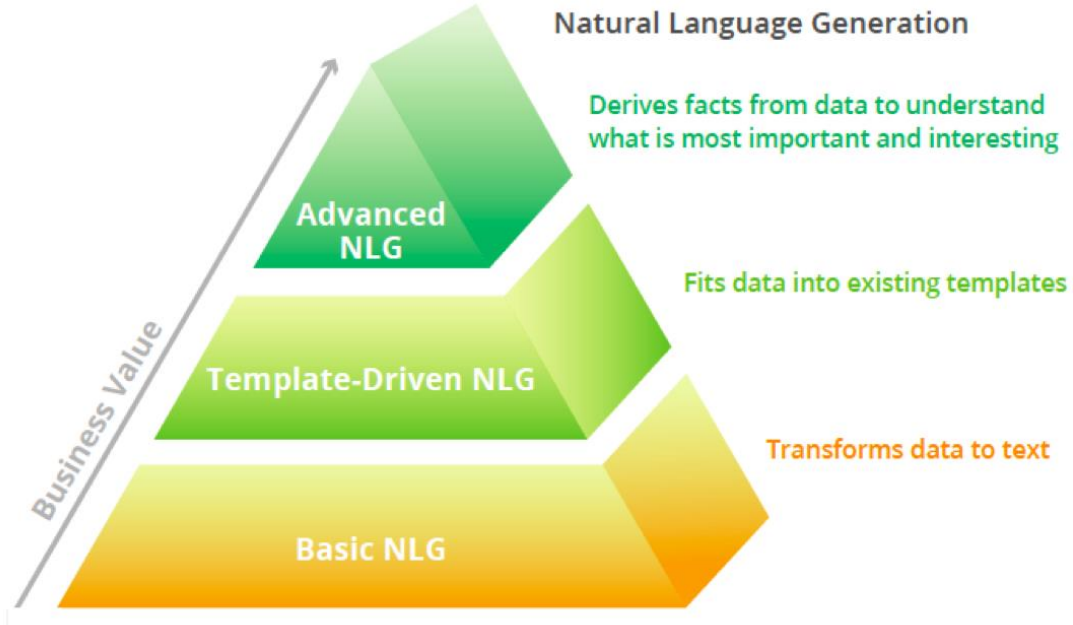


NL Understanding

Natural Language Understanding (NLU) helps the machine to understand and analyze human language by extracting the text from large data such as keywords, emotions, relations, and semantics, etc.



NL Generation



Natural Language Generation(NLG) is extracting meaningful insights as phrases and sentences in the form of natural language.

It consists –

- **Text planning –** It includes retrieving the relevant data from the domain.
- **Sentence planning –** It is nothing but a selection of important words, meaningful phrases, or sentences.

Natural language understanding

Raw speech signal

↓ • **Speech recognition**

Sequence of words spoken

↓ • **Syntactic analysis** using knowledge of the grammar

Structure of the sentence

↓ • **Semantic analysis** using info. about meaning of words

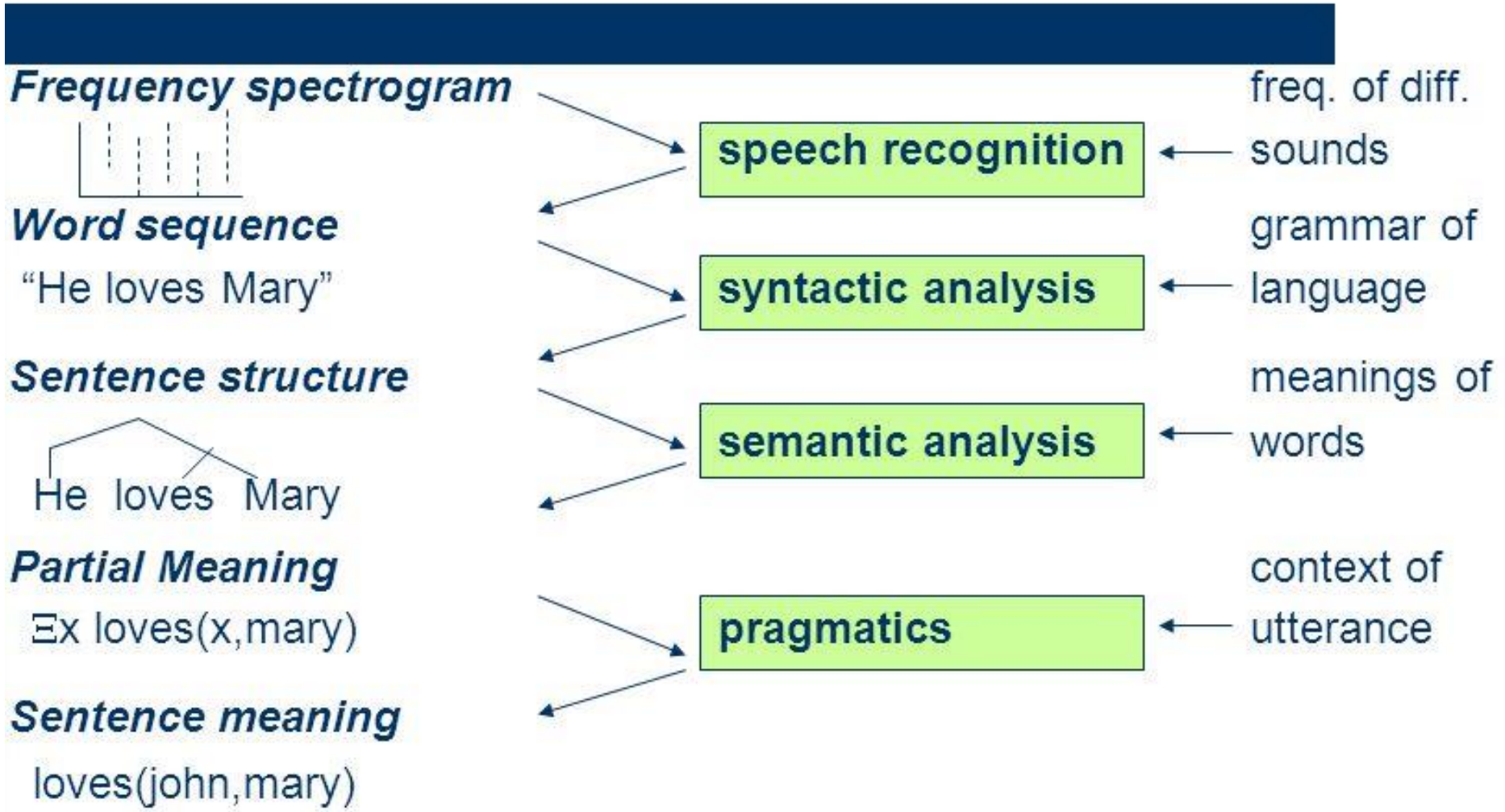
Partial representation of meaning of sentence

↓ • **Pragmatic analysis** using info. about context

Final representation of meaning of sentence

Natural Language Understanding

- Input/Output data Processing stage Other data used



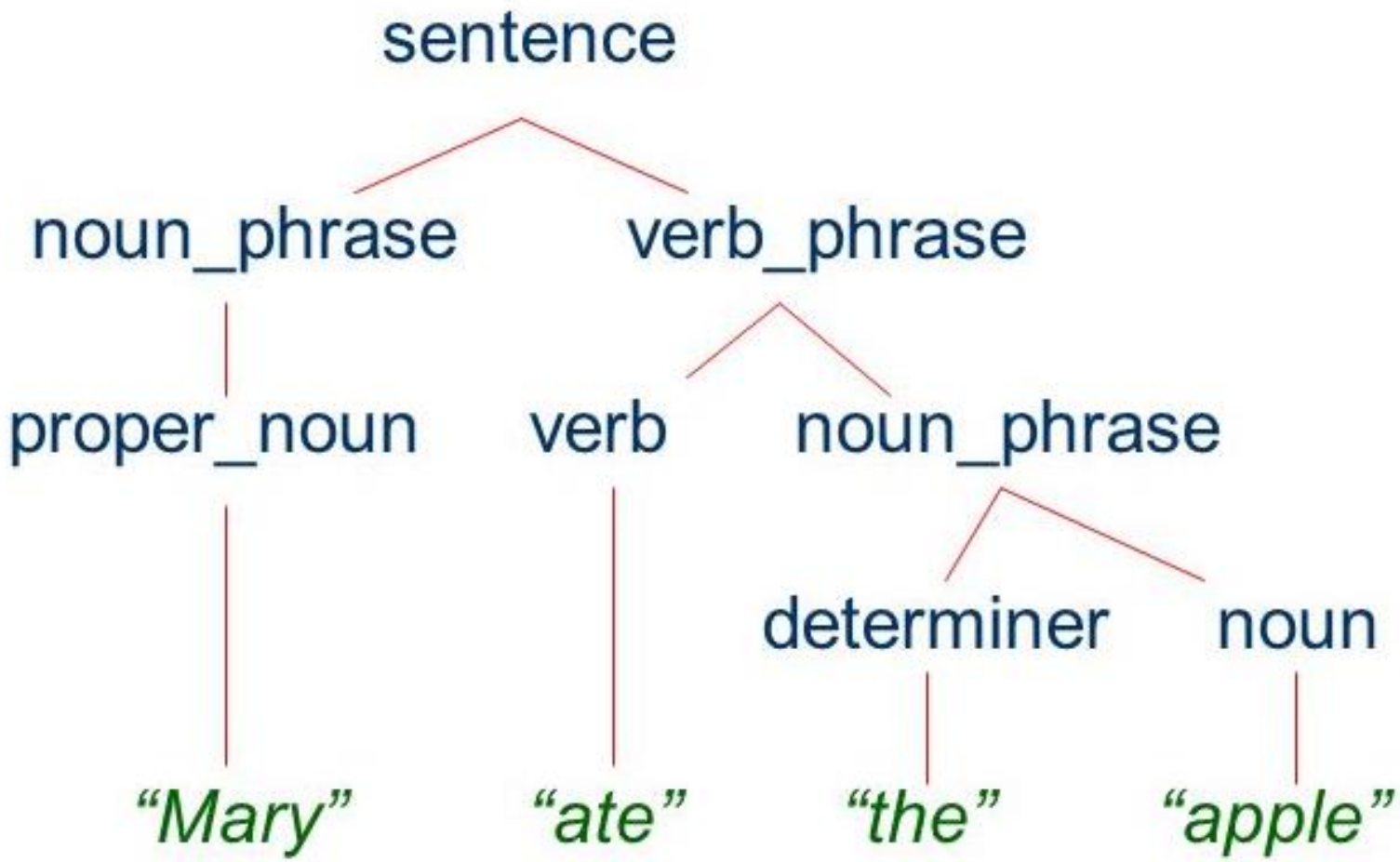
Syntactic Analysis

- Rules of syntax (grammar) specify the possible organization of words in sentences and allows us to determine sentence's structure(s)
  - "John saw Mary with a telescope"
    - John saw (Mary with a telescope)
    - John (saw Mary with a telescope)
- Parsing: given a sentence and a grammar
  - Checks that the sentence is correct according with the grammar and if so returns a **parse tree** representing the structure of the sentence

Syntactic Analysis - Grammar

- sentence -> noun_phrase, verb_phrase
- noun_phrase -> proper_noun
- noun_phrase -> determiner, noun
- verb_phrase -> verb, noun_phrase
- proper_noun -> [mary]
- noun -> [apple]
- verb -> [ate]
- determiner -> [the]

Syntactic Analysis - Parsing



Syntactic Analysis – Complications (1)

- Number (singular vs. plural) and gender
  - `sentence-> noun_phrase,verb_phrase`
  - `proper_noun(s) -> [mary]`
  - `noun(p) -> [apples]`
- Adjective
  - `noun_phrase-> determiner,adjectives,noun`
  - adjectives-> adjective, adjectives
  - adjective->[ferocious]
- Adverbs, …

Syntactic Analysis – Complications (2)

- Handling ambiguity
  - Syntactic ambiguity: “fruit flies like a banana”
- Having to parse syntactically incorrect sentences

Semantic Analysis – Complications

- Handling ambiguity
  - Semantic ambiguity: "I saw the prudential building flying into Boston"

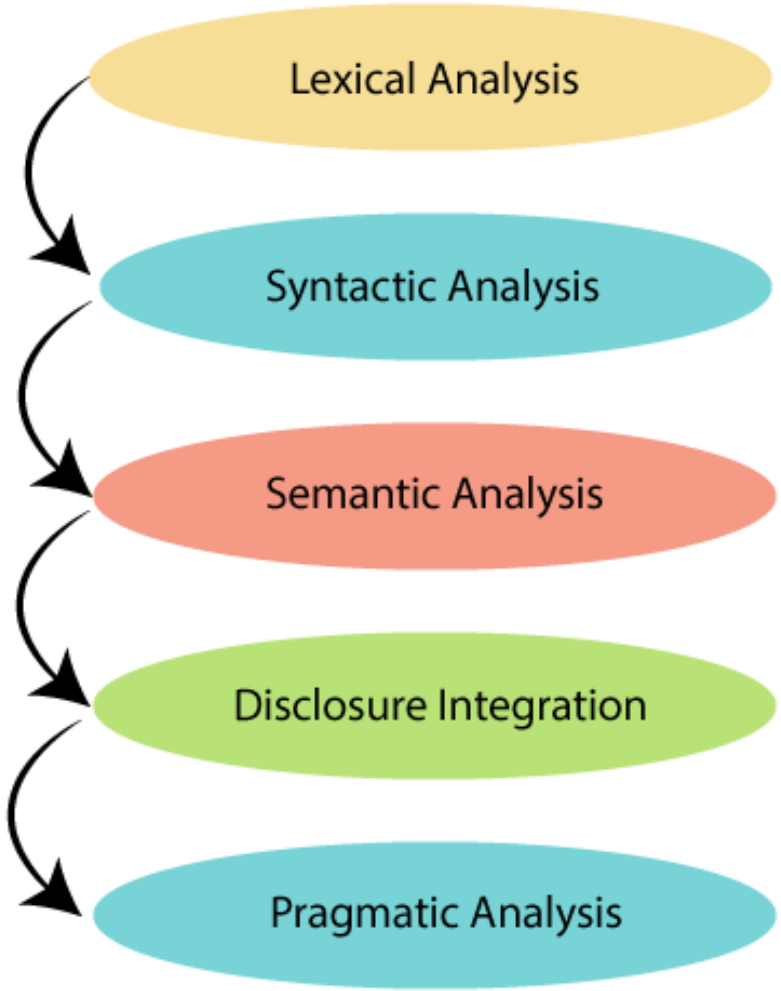
Pragmatics

- Uses context of utterance
  - Where, by who, to whom, why, when it was said
  - Intentions: *inform, request, promise*, *criticize,* …
- Handling Pronouns
  - "Mary eats apples. She likes them."
    - She="Mary", them="apples".
- Handling ambiguity
  - Pragmatic ambiguity: "you're late": What's the speaker's intention: informing or criticizing?

Natural Language Generation

- Talking back! ☺
- What to say or text planning
  - flight(AA,london,boston,$560,2pm),
  - flight(BA,london,boston,$640,10am),
- How to say it
  - "There are two flights from London to Boston. The first one is with American Airlines, leaves at 2 pm, and costs $560 …"
- Speech synthesis
  - Simple: Human recordings of basic templates
  - More complex: string together phonemes in phonetic spelling of each word
    - Difficult due to stress, intonation, timing, liaisons between words

Phases of NLP



Lexical Analysis Phase

Lexical Analysis:

- It involves identifying and analyzing the structure of words.
- Lexicon of a language means the collection of words and phrases in that particular language.
- The lexical analysis divides the text into paragraphs, sentences, and words.

So we need to perform Lexicon Normalization.

The most common lexicon normalization techniques are:

•**Stemming:** Stemming is the process of reducing derived words to their word stem, base, or root form—generally a written word form like-“ing”, “ly”, “es”, “s”, etc.

•**Lemmatization:** Lemmatization is the process of reducing a group of words into their lemma or dictionary form. It takes into account things like POS(Parts of Speech), the meaning of the word in the sentence, the meaning of the word in the nearby sentences, etc. before reducing the word to its lemma.

Syntactic Analysis Phase

Syntactic Analysis is used to check grammar, arrangements of words, and the interrelationship between the words.

Example: Mumbai goes to the Sara

Here “Mumbai goes to Sara”, which does not make any sense, so this sentence is rejected by the Syntactic analyzer.
Syntactical parsing involves the analysis of words in the sentence for grammar. Dependency Grammar and Part of Speech (POS)tags are the important attributes of text syntactic.

Semantic Analysis Phase

Retrieves the possible meanings of a sentence that is clear and semantically correct.

Its process of retrieving meaningful insights from text.

Discourse Integration Phase

- It is nothing but a sense of context. That is sentence or word depends upon that sentences or words.
- It's like the use of proper nouns/pronouns.

For example, Ram wants it.

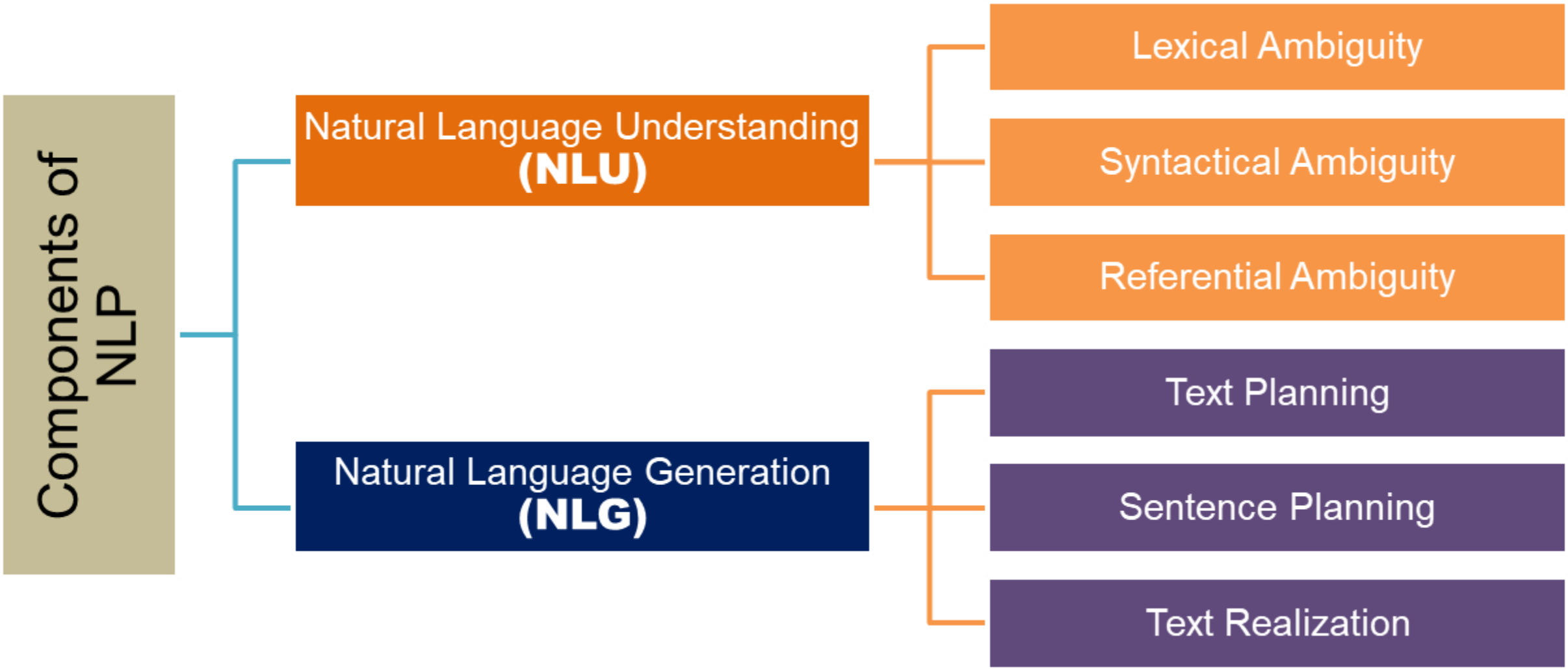
- In the above statement, we can clearly see that the "it" keyword does not make any sense.
- In fact, it is referring to anything that we don't know.

That is nothing but this "it" word depends upon the previous sentence which is not given. So once we get to know about "it", we can easily find out the reference.

Pragmatic Analysis Phase

- It means the study of meanings in a given language.
- Process of extraction of insights from the text.
- It includes the repetition of words, who said to whom? etc.

It understands that how people communicate with each other, in which context they are talking and so many aspects.


Components of NLP
Natural Language Understanding (NLU)
Lexical Ambiguity
Syntactical Ambiguity
Referential Ambiguity
Natural Language Generation (NLG)
Text Planning
Sentence Planning
Text Realization

Exercises

1. He is looking for a match.

What do you understand by the 'match' keyword? Does it partner or cricket or football or anything else?
This is **Lexical Ambiguity**. It happens when a word has different meanings. Lexical ambiguity can be resolved by using parts-of-speech (POS)tagging techniques.

2. The Fish is ready to eat.

What do you understand by the above example? Is the fish ready to eat his/her food or fish is ready for someone to eat? Got confused!! Right?
This is ***Syntactical Ambiguity*** which means when we see more meanings in a sequence of words and also Called Grammatical Ambiguity.

3. Chirag met Pranav and Dinesh. They went to a restaurant.

Here, they refer to Pranav and Dinesh or all.
Referential Ambiguity: It is very often in a text that it mentions an entity (something/someone), and then refers to it again, possibly in a different sentence, with the help of another word. So, these different pronouns can cause ambiguity when it is not clear which noun it is referring to.

Exercises

Identify various interpretations of the given sentences.

Sentence: I saw a student on a hill with a microscope.

These are various interpretations of the above sentence which are shown below:
- There is a student on the hill, and I watched him with my microscope.
- There is a student on the hill, and he has a microscope.
- I'm on a hill, and I saw a student using my microscope.
- I'm on a hill, and I saw a student who has a microscope.
- There is a student on a hill, and I saw him something with my microscope.

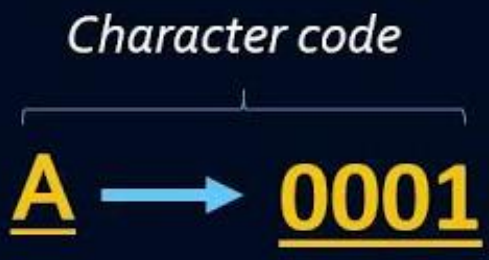
Sentence: Can you help me with the can?

In the sentence above, we observed that there are two "can" words, but they have different meanings.
Here.
The first "can" word is used to form a question.
The second "can" word that is used at the end of the sentence is used to represent a container that holds some things such as food or liquid, etc.

From the examples, we can observe that language processing is not “deterministic” that is the same language has the same interpretations, and something suitable to one person might not be suitable to another person.
Therefore, Natural Language Processing (NLP) has a non-deterministic approach.

We can use Natural Language Processing to create a new intelligent or AI system that can understand in the same way as that of humans and interpret the language in different situations.

Representing Text



- **Characters are encoded with the use of character sets**. These are codes that assign a character to a unique bit pattern e.g.:
- This means text can be represented by long strings of 0s and 1s.
- In the early days of computing, many different character sets were created, which caused **compatibility issues**. In an attempt to standardise this, **ASCII** (**American Standard Code for Information Interchange**) was created.
- The original ASCII character set represents characters with 7 bits each, but then it was extended to have a bit length of 8. This allows 128 and 256 characters to be represented, respectively.
- However, 256 is **not sufficient to represent all possible characters**, so a new standard called **Unicode** was developed. ASCII is a sub-set of Unicode (so they share character codes)
- Unicode used 16 bits for each character, but this has since been expanded. It currently has codes for 120,000+ characters. It aims to cover all symbols and characters ever used.

Text Encoding

1. Machine doesn't understand characters, words or sentences.
2. Machines can only process numbers.
3. Text data must be encoded as numbers for input or output for any machine.

WHY to perform text encoding?
As mentioned in the above points we cannot pass raw text into machines as input until and unless we convert them into numbers, hence perform text encoding.

WHAT is text encoding?
Text encoding is a process to convert meaningful text into number / vector representation so as to preserve the context and relationship between words and sentences, such that a machine can understand the pattern associated in any text and can make out the context of sentences.

Text Encoding Schemes

HOW to encode text for any NLP task?

There are a lot of methods to convert Text into numerical vectors, such as:

- Index-Based Encoding
- Bag of Words (BOW)
- TF-IDF Encoding (Term Frequency — Inverse Document Frequency)
- Word2Vector Encoding

Document Corpus: This is the whole set of text we have, basically our text corpus, can be anything like news articles, blogs, etc.…

Example: We have 5 sentences namely,
[“this is a good phone” , “this is a bad mobile” , “she is a good cat” , “he has a bad temper” , “this mobile phone is not good”]

Data Corpus: It is the collection of unique words in our document corpus, i.e. in our case it looks like this:

[“a” , “bad” , “cat” , “good” , “has” , “he” , “is” , “mobile” , “not” , “phone” , “she” , “temper” , “this”]

Index-Based Text Encoding

1. Index-Based Encoding:

As the name mentions, in Index based, need to give all the unique words an index, like we have separated out our Data Corpus, now can index them individually, like

a : 1

bad : 2

…

this : 13

Now that we have assigned a unique index to all the words so that based on the index we can uniquely identify them and can convert sentences using this index-based method.

Just replacing the words in each sentence with their respective indexes.

Our document corpus becomes:
[13 7 1 4 10] , [13 7 1 2 8] , [11 7 1 4 3] , [6 5 1 2 12] , [13 8 10 7 9 4]

Now we have encoded all the words with index numbers, and this can be used as input to any machine since machine understands number.

But there is a tiny bit of issue which needs to be addressed first and that is the consistency of the input.
Our input needs to be of the same length as our model, it cannot vary.

It might vary in the real world but needs to be taken care of when we are using it as input to our model.

Now as we can see the first sentence has 5 words, but the last sentence has 6 words, this will cause an imbalance in our model.

So to take care of that issue, do max padding, which means take the longest sentence from our document corpus and pad the other sentence to be as long.

This means if all of the sentences are of 5 words and one sentence is of 6 words, then make all the sentences of 6 words.

Now how do we add that extra word here?

In our case how do we add that extra index here?

If you have noticed we didn't use 0 as an index number, and preferably that will not be used anywhere even if we have 100000 words long data corpus, hence use 0 as the padding index. This also means that we are appending nothing to our actual sentence as 0 doesn't represent any specific word, hence the integrity of our sentences are intact.

So finally our index based encodings are as follows:

[13 7 1 4 10 0] ,

[13 7 1 2 8 0] ,

[11 7 1 4 3 0] ,

[6 5 1 2 12 0] ,

[13 8 10 7 9 4]

And this is how our input's integrity are same and without disturbing the context of our sentences as well.

Index-Based Encoding considers the sequence information in text encoding.

Bag of Words (BOW)

2. Bag of Words (BOW):

Bag of Words or BoW is another form of encoding where we use the whole data corpus to encode our sentences. It will make sense once we see actually how to do it.

Data Corpus:

[“a” , “bad” , “cat” , “good” , “has” , “he” , “is” , “mobile” , “not” , “phone” , “she” , “temper” , “this”]

As we know that our data corpus will never change, so if we use this as a baseline to create encodings for our sentences, then we will be on an upper hand to not pad any extra words.

Now, 1st sentence we have is this : “this is a good phone”

How do we use the whole corpus to represent this sentence?

a	bad	cat	good	has	he	is	mobile	not	phone	she	temper	this
1	0	0	1	0	0	1	0	0	1	0	0	1

So our first sentence becomes a combination of all the words we have.

[1,0,0,1,0,0,1,0,0,1,0,0,1]

This is how our first sentence is represented.

Types of Bag of Words (BOW)

1. Binary BOW
2. BOW

The difference between them is, in Binary BOW encode 1 or 0 for each word appearing or non-appearing in the sentence. We do not take into consideration the frequency of the word appearing in that sentence.

In BOW we also take into consideration the frequency of each word occurring in that sentence.
Let's say our text sentence is "this is a good phone this is a good mobile"

Binary BOW : [1,0,0,1,0,0,1,1,0,1,0,0,1]

a	bad	cat	good	has	he	is	mobile	not	phone	she	temper	this
1	0	0	1	0	0	1	1	0	1	0	0	1

BOW : [2,0,0,2,0,0,2,1,0,1,0,0,2]

a	bad	cat	good	has	he	is	mobile	not	phone	she	temper	this
2	0	0	2	0	0	2	1	0	1	0	0	2

If you see carefully, we considered the number of times the words "this", "a", "is" and "good" have occurred.
So that's the only difference between Binary BOW and BOW.
BOW totally discards the sequence information of our sentences.

TF-IDF Encoding

3. TF-IDF Encoding:

Term Frequency — Inverse Document Frequency

As the name suggests, here we give every word a relative frequency coding w.r.t the current sentence and the whole document.

Term Frequency: Is the occurrence of the current word in the current sentence w.r.t the total number of words in the current sentence.

Inverse Data Frequency: Log of Total number of words in the whole data corpus w.r.t the total number of sentences containing the current word.

TF:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}}$$

IDF:

$$idf(w) = log(\frac{N}{df_t})$$

One thing to note here is we have to calculate the word frequency of each word for that particular sentence, because depending on the number of times a word occurs in a sentence the TF value can change, whereas the IDF value remains constant, until and unless new sentences are getting added.
Let's try to understand by experimenting it out:
Data Corpus: ["a" , "bad" , "cat" , "good" , "has" , "he" , "is" , "mobile" , "not" , "phone" , "she" , "temper" , "this"]

TF-IDF : "this" in sentence1 : Number of "this" word in sentence1 / total number of words in sentence1
IDF : log(total number of words in the whole data corpus / total number of sentences having "this" word)
TF : 1 / 5 = 0.2
IDF : log (13 / 3) = 0.6368
TF-IDF : 0.2 * 0.6368 = 0.127

So we associate “this” : 0.127

similarly we can find out TF-IDF for every word in that sentence and then rest of the process remains same as BOW, here we replace the word not with the frequency of its occurrence but rather with the TF-IDF value for that word.

So let’s try to encode our first sentence: “this is a good phone”

Note: In this example natural logarithm is used for computation

a	bad	cat	good	has	he	is	mobile	not	phone	she	temper	this
0.2593	0	0	0.3226	0	0	0.2593	0	0	0.4123	0	0	0.3226

As we can see that we have replaced all the words appearing in that sentence with their respective tf-idf values, one thing to notice here is, similar tf-idf values of multiple words are present.

This is a rare case that has happened with us as we had few documents and almost all words had kind of similar frequencies.

And this is how we achieve TF-IDF encoding of text.

Now try to implement them:

```
document_corpus = ["this is a good phone phone" ,
                   "this is a bad mobile mobile" ,
                   "she is a good good cat" ,
                   "he has a bad temper temper" ,
                   "this mobile phone phone is not good good"]
```

This is our document corpus as mentioned above.

```
data_corpus = set()
for row in document_corpus:
    for word in row.split(" "):
        if word not in data_corpus:
            data_corpus.add(word)

data_corpus=sorted(data_corpus)

print(data_corpus)
```

```
['a', 'bad', 'cat', 'good', 'has', 'he', 'is', 'mobile', 'not', 'phone', 'she', 'temper', 'this']
```

This is how we are creating our data corpus based on any document corpus.

Index Based Encoding :

```
res = len(max(document_corpus, key = len).split(" "))
print(res)
```

```
8
```

```
index_based_encoding=[]
for row in document_corpus:
    row_encoding = []
    split = row.split(" ")
    for i in range(res):
        if i <= len(split)-1:
            row_encoding.append(data_corpus.index(split[i])+1)
        else:
            row_encoding.append(0)
    index_based_encoding.append(row_encoding)

print(index_based_encoding)
```

```
[[13, 7, 1, 4, 10, 10, 0, 0], [13, 7, 1, 2, 8, 8, 0, 0], [11, 7, 1, 4, 4, 3, 0, 0], [6, 5, 1, 2, 12, 12, 0, 0], [13, 8, 10, 10,
7, 9, 4, 4]]
```

First we find out the max length of a sentence and then using our data corpus we encode all our text with index based scheme.

Bag Of Words (BoW) :

1. Binary BoW

```
]: one_hot_encoding = []
for row in document_corpus:
    row_encoding = []
    split = row.split(" ")
    for word in data_corpus:
        if word in split:
            row_encoding.append(1)
        else:
            row_encoding.append(0)
    one_hot_encoding.append(row_encoding)

print(one_hot_encoding)
```

```
[[1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1], [1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1], [1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0], [1,
1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0], [0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1]]
```

Implementing Binary BoW, where we place 1 for every word encountered in the sentence in the data corpus and 0 for the rest.

2. BoW

```
one_hot_encoding = []
for row in document_corpus:
    row_encoding = []
    split = row.split(" ")
    for word in data_corpus:
        count = split.count(word)
        if word in split:
            row_encoding.append(count)
        else:
            row_encoding.append(count)
    one_hot_encoding.append(row_encoding)

print(one_hot_encoding)
```

```
[[1, 0, 0, 1, 0, 0, 1, 0, 0, 2, 0, 0, 1], [1, 1, 0, 0, 0, 0, 1, 2, 0, 0, 0, 0, 1], [1, 0, 1, 2, 0, 0, 1, 0, 0, 0, 1, 0, 0], [1,
1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 2, 0], [0, 0, 0, 2, 0, 0, 1, 1, 1, 2, 0, 0, 1]]
```

Implementing BoW, here we have to encode the count of each occurrences of the word in that particular sentence to the data corpus, and 0 for the rest.

TF-IDF Encoding :

```
tf_dict = {}
i=0
for row in document_corpus:
    row_dict={}
    split = row.split(" ")
    for word in split:
        if word not in row_dict.keys():
            row_dict[word] = split.count(word)
    tf_dict[i] = row_dict
    i+=1

print(tf_dict)
```

```
{0: {'this': 1, 'is': 1, 'a': 1, 'good': 1, 'phone': 2}, 1: {'this': 1, 'is': 1, 'a': 1, 'bad': 1, 'mobile': 2}, 2: {'she': 1,
'is': 1, 'a': 1, 'good': 2, 'cat': 1}, 3: {'he': 1, 'has': 1, 'a': 1, 'bad': 1, 'temper': 2}, 4: {'this': 1, 'mobile': 1, 'phon
e': 2, 'is': 1, 'not': 1, 'good': 2}}
```

First calculating frequency of every element w.r.t the sentences

```
import math
def calculate_tf(word, sentence_num):
    row_dict = tf_dict[int(sentence_num)]
    return row_dict[word]/sum(row_dict.values())

def calculate_idf(word):
    doc_num = 0
    for key, value in tf_dict.items():
        if word in value.keys():
            doc_num+=1
    return math.log(len(data_corpus)/doc_num+1)

def tf_idf(word, sentence_num):
    return round(calculate_tf(word, sentence_num) * calculate_idf(word),5)
```

```
tf_idf('phone',0)
```

```
0.7167
```

Creating function to calculate tf-idf for every word in a particular sentence, which takes reference of the above created frequencies

```
tf_idf_encoding = []
for i in range(len(document_corpus)):
    row = document_corpus[i]
    split = row.split(" ")
    row_encoding = []
    for word in data_corpus:
        if word in split:
            row_encoding.append(tf_idf(word,i))
        else:
            row_encoding.append(0)
    tf_idf_encoding.append(row_encoding)

print(tf_idf_encoding)
```

```
[[0, 0, 0.27726, 0, 0, 0.21972, 0, 0, 0.7167, 0, 0, 0.27726], [0.35835, 0, 0, 0, 0, 0.21972, 0.7167, 0, 0, 0, 0, 0.27726], [0,
0.49698, 0.55452, 0, 0, 0.21972, 0, 0, 0, 0.49698, 0, 0], [0.35835, 0, 0, 0.49698, 0.49698, 0, 0, 0, 0, 0, 0.99396, 0], [0, 0,
0.34657, 0, 0, 0.13733, 0.22397, 0.31061, 0.44794, 0, 0, 0.17329]]
```

Finally creating the tf-idf vectors for our module.

GitHub link **here**

Practice problems for Text Encoding

Given the following sentences, apply the different encoding techniques.

1.He is eating Veg

2. She is eating NonVeg

3.Both are eating Food

Word2Vec Text Encoding

Word2Vec is not a singular algorithm, rather, it is a family of model architectures and optimizations that can be used to learn word embeddings from large datasets.

Embeddings learned through Word2Vec have proven to be successful on a variety of downstream natural language processing tasks.

Tools are available to perform text encoding: An Example

"word2vec" tool provides an efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representations of words.
These representations can be subsequently used in many natural language processing applications and for further research.

Quick start

- Download the code: svn checkout http://word2vec.googlecode.com/svn/trunk/
- Run 'make' to compile word2vec tool
- Run the demo scripts: **./demo-word.sh** and **./demo-phrases.sh**
- For questions about the toolkit, see http://groups.google.com/group/word2vec-toolkit

Thank You