NETZWERK DATA SCIENCE ACADEMY

Natural Language Processing Training

Made By – AMIT DHOMNE

Software

- Anaconda Jupyter
- Python 3.6 or 2.7
- Install various libraries NLTK, sPacy, Sklearn ,pandas,numpy, tensor flow etc.

Introduction

Natural-language processing (NLP) is an area of computer science and artificial intelligence concerned with the **interactions** between computers and human (natural) languages

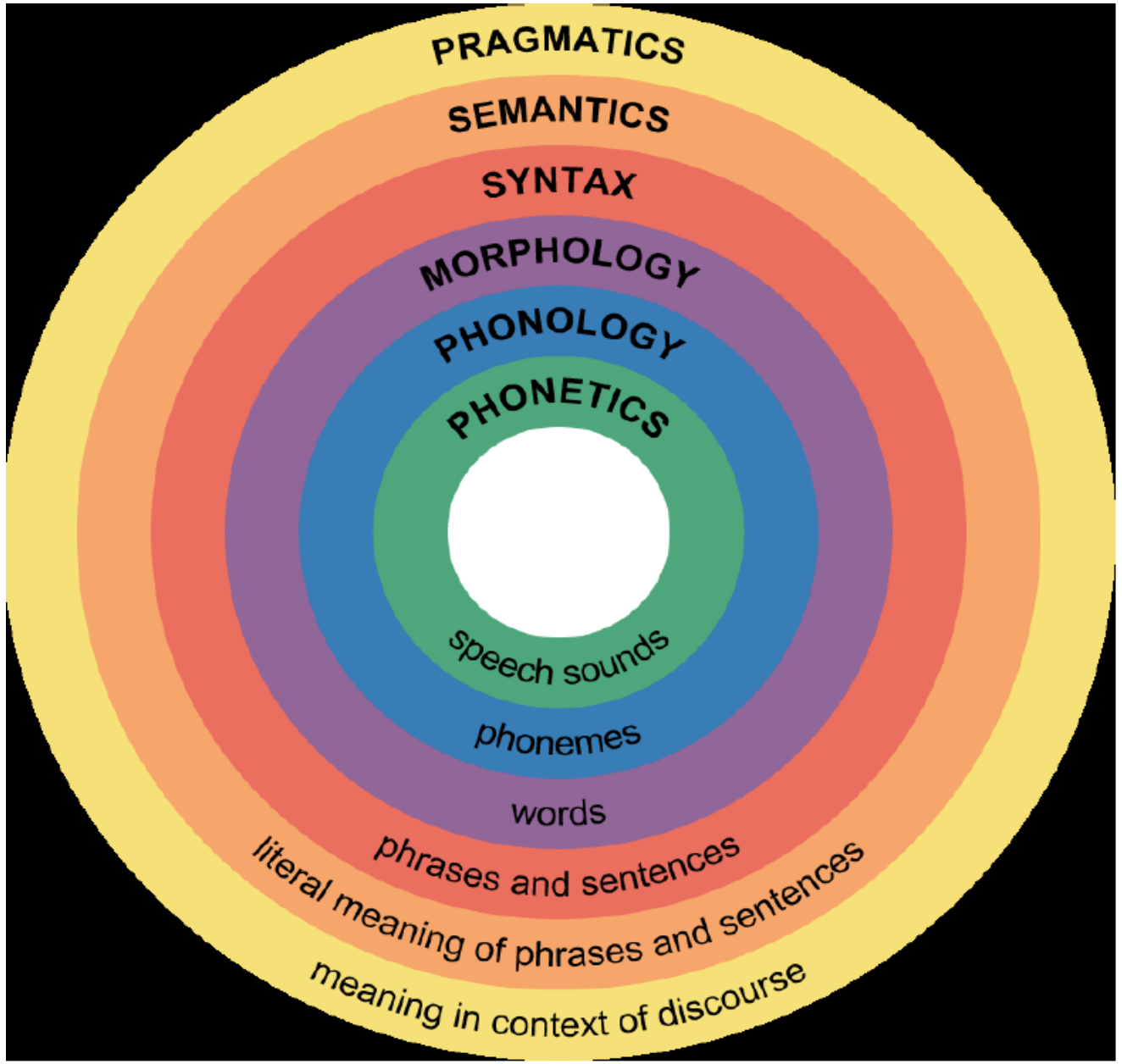
In particular, is concerned with how to program computers to fruitfully **process** large amounts of natural language data.

Challenges in NPL

frequently involve speech recognition, natural-language understanding, and natural-language generation

NLP is **characterized** as a hard problem in computer science as human language is rarely precise or plainly spoken

Understand semantics - --- **apply** your knowledge of the physical world , **Context is everything in NLP**



Challenges:

- Variability
- Ambiguity
- Meaning is context dependent
- Requires background knowledge

Ref: CSE 628 - Introduction to NLP (Professor Niranjan Balasubramanian)
Image From: Commons.wikimedia.org

Explanation

How does the communication context affect meaning?

What are the meanings of words, phrases etc.?

How do words form phrases, and phrases sentences?

How do morphemes, i.e. sub-word units, form words?

How do phonemes, i.e., sound units, form pronunciations?

How are the speech sounds generated and perceived?

How Machine Interprets Text and Meaning

- How does the communication context affect meaning?
- What are the meanings of words, phrases etc.?
- How do words form phrases, and phrases sentences?
- How do morphemes, i.e. **sub-word** units, form words?
- How do phonemes, i.e., **sound units**, form pronunciations?
- How are the speech sounds generated and perceived?

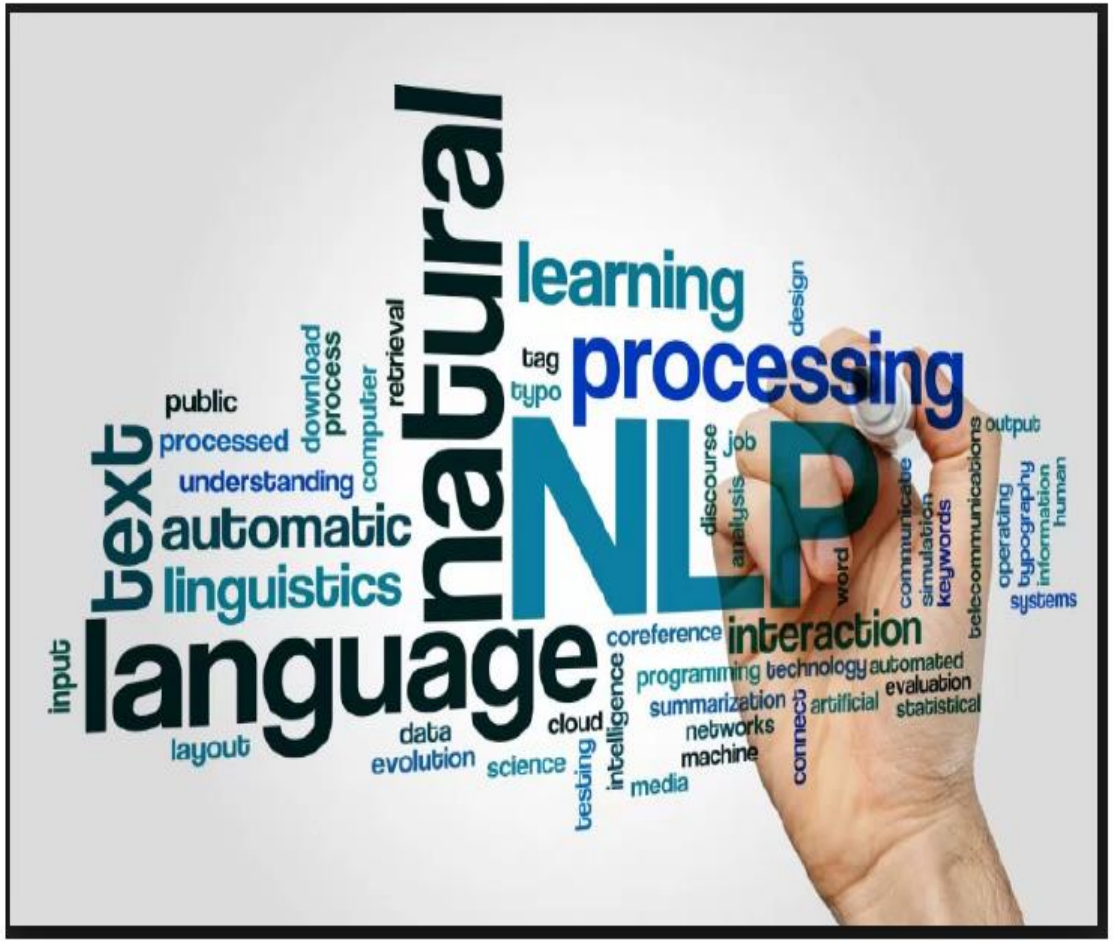
Some NLP applications

1. Spelling and Grammar Correction/ detection (Eg. MS-Word, Grammarly etc.)

2. Machine Translation (Eg. Google Translate, Bing Translate)

3. Opinion Mining (Eg. Extract sentiment of demographic from blogs and social media)

4. Speech Recognition and Synthesis (Eg. Siri, Google assistant, Amazon Alexa)



NLP Toolkits

- spaCy (Python)
- NLTK (Python)

NLP Goal (Pre processing)

- What is preprocessing – It means remove all extra having less important words, remove stop words, tokenization (regexp) , word normalization
- dog ~ dogs
- .tolower(), regexp, stemming, lemmatization
- Stop words example :

```
> stopwords("english")
  [1] "i"          "me"         "my"         "myself"     "we"
  [6] "our"        "ours"       "ourselves"  "you"        "your"
 [11] "yours"      "yourself"   "yourselves" "he"         "him"
 [16] "his"        "himself"    "she"        "her"        "hers"
 [21] "herself"    "it"         "its"        "itself"     "they"
 [26] "them"       "their"      "theirs"     "themselves" "what"
 [31] "which"      "who"        "whom"       "this"       "that"
 [36] "these"      "those"      "am"         "is"         "are"
 [41] "was"        "were"       "be"         "been"       "being"
 [46] "have"       "has"        "had"        "having"     "do"
```

NLP Goal (Pre processing) (Continue)

- cleanup, tokenization
- stemming
- lemmatization
- part-of-speech tagging
- query expansion
- sentence segmentation
- optical character recognition (OCR)
- speech processing
- speech recognition
- text-to-speech
- information extraction
- named entity recognition (NER)
- sentiment analysis
- word sense disambiguation
- text similarity

Word, term, feature

- word <> term
- **document** or text chunk is an **unit / entity / object!**
- **terms** are **features** of the document**!**
- each term has properties:
- normalized form -> term.baseform + term.transformation
- position(s) in the document -> term.position(s)
- frequency -> term.frequency

Text, document, chunk

- what is document?
- text segmentation
- hard problem
- usually we consider whole document as one **unit (entity)**
- **Text –** Collection of character
- **Chunk** – Collection of texts, words
- **Document** - Collection of words, text, sentence etc

NLP Algorithm

- Naive Bayes
- Markov models
- SVM
- Neural networks / Deep learning

Vector Representations of Text for Machine Learning

- In order to put the words into the machine learning algorithm the text data should be converted into a vector representations.
- There are three major ways of doing that.

One-Hot Vectors

- **Machine** learning algorithms work with numeric values and NLP **application** generate data in the form of words and sentences
- One way to **convert** the words into numeric values is **one-hot vectors**
- We take all the words that are present in the dictionary and **make vectors** such that one index represents the word and the rest all are zeros

Problem with One-Hot Vectors

- Machine Learning algorithms using
- One-Hot Vectors are computationally expensive

- They do not consider the similarity between words

How do we convert words to numbers?

One −Hot VECTORS


```
Rome   = [1, 0, 0, 0, 0, 0, …, 0]

Paris  = [0, 1, 0, 0, 0, 0, …, 0]

Italy  = [0, 0, 1, 0, 0, 0, …, 0]

France = [0, 0, 0, 1, 0, 0, …, 0]
```

Term weight - TF-IDF I am amit

- term frequency – inverse document frequency
- variables:
- t - term,
- d - one document
- D - all documents

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \times \mathrm{idf}(t, D)$$

- TF - is term frequency in a document function - i.e. measure on how much information the term brings in **one document**
- IDF - is inverse document frequency of the term function - i.e.
- inversed measure on how much information the term brings in **all documents** (corpus)

Consider a document containing 100 words where the word *cat* appears 3 times. The term frequency (i.e., tf) for *cat* is then (3 / 100) = 0.03. Now, assume we have 10 million documents and the word *cat* appears in one thousand of these. Then, the inverse document frequency (i.e., idf) is calculated as log(10,000,000 / 1,000) = 4. Thus, the Tf-idf weight is the product of these quantities: 0.03 * 4 = 0.12. note TF * IDF = 0.12 , 1 0 1

- TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).
- IDF(t) = log_e(Total number of documents / Number of documents with term t in it).
- t — term (word) ,d — document (set of words) ,N — count of corpus
- corpus — the total document set // collection of document ,10 files,12
- *tf(t,d) = count of t in d / number of words in d*
- *df(t) = occurrence of t in documents*
- *idf(t) = N/df : N = total of document=10*
- *idf(t) = log(N/(df + 1)) :0.0000012 <<<0*

Here A and B are the sentences which is having this words in word column

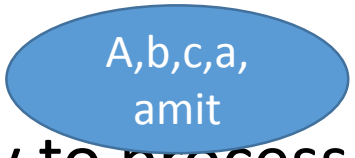
Word	TF		IDF	TF*IDF	
	A	B		A	B
The	1/7	1/7	log(2/2) = 0	0	0
Car	1/7	0	log(2/1) = 0.3	0.043	0
Truck	0	1/7	log(2/1) = 0.3	0	0.043
Is	1/7	1/7	log(2/2) = 0	0	0
Driven	1/7	1/7	log(2/2) = 0	0	0
On	1/7	1/7	log(2/2) = 0	0	0
The	1/7	1/7	log(2/2) = 0	0	0
Road	1/7	0	log(2/1) = 0.3	0.043	0
Highway	0	1/7	log(2/1) = 0.3	0	0.043

Tf-IDF Disadvantage

- TF-IDF is based on the bag-of-words (BoW) model, therefore it does not capture position in text, semantics, co-occurrences in different documents, etc.
- For this reason, TF-IDF is only useful as a lexical level feature
- Cannot capture semantics (e.g. as compared to topic models, word embeddings)

Word Embedding

Bag of words /// a= 2, b



- simplified and effective way to process documents by:
- disregarding grammar (term.baseform?)
- disregarding word order (term.position)
- keeping only multiplicity (term.frequency)
- we take a document and find out the frequencies of occurrence of words in it
- And then these frequencies are fed into the machine learning algorithm

Bag of Words Example

Document 1

The quick brown fox jumped over the lazy dog's back.

Document 2

Now is the time for all good men to come to the aid of their party.

Term	Document 1	Document 2
aid	0	1
all	0	1
back	1	0
brown	1	0
come	0	1
dog	1	0
fox	1	0
good	0	1
jump	1	0
lazy	1	0
men	0	1
now	0	1
over	1	0
party	0	1
quick	1	0
their	0	1
time	0	1

Stopword List

for
is
of
the
to

Bag-of-words


```
doc_1 = [32, 14, 1,  0, …,  6]

doc_2 = [ 2, 12, 0, 28, …, 12]

…      …

doc_N = [13,  0, 6,  2, …, 0]
```

Bag of Word Model

- **N-gram**

Problem with Bag-of-Words

- Count of word is important
- Too simplistic
- Ignores the context of the word
- Loses the ordering of the words
- **For example:** “My name is John” is same as “Is my name John?”

STEMMING

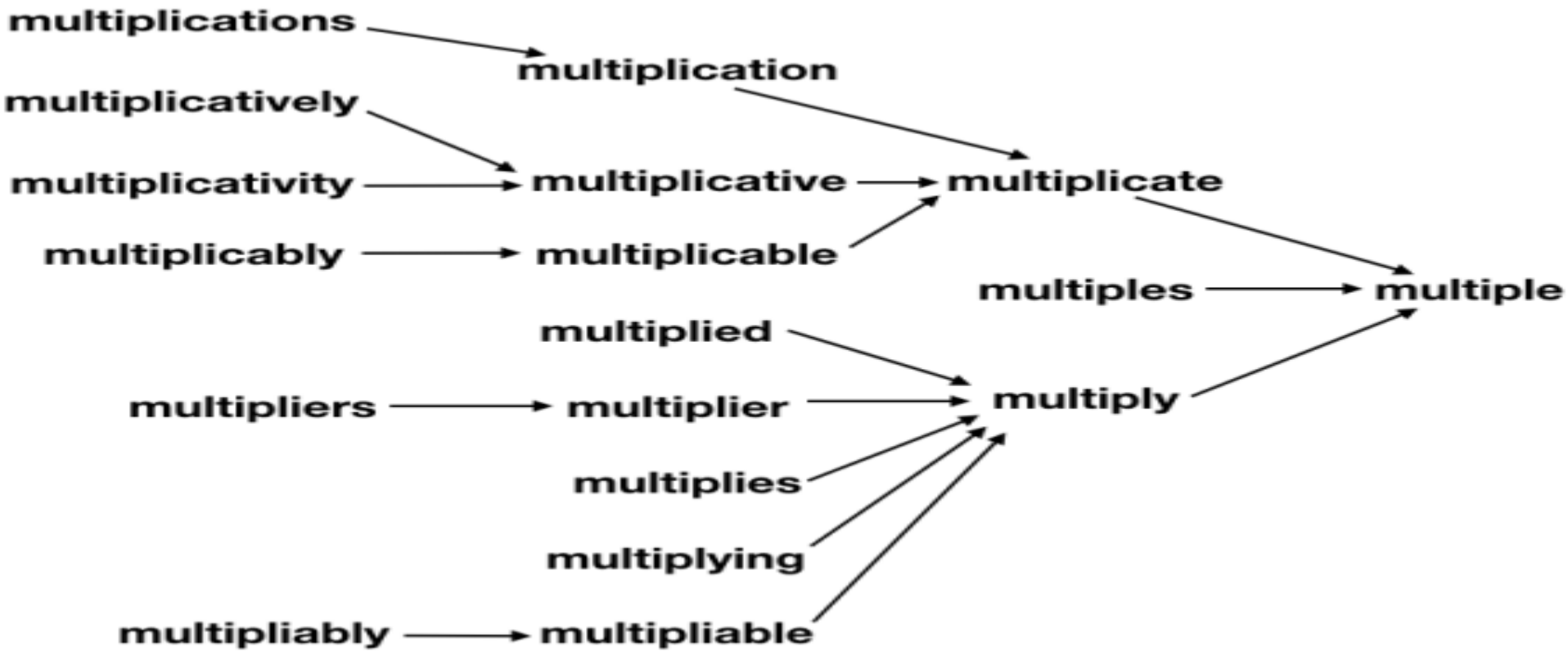
am, are, is → be

Car cars, car's, cars' → car

Using above mapping a sentence could be normalized as follows:

the boy's cars are different colors → the boy car be differ color

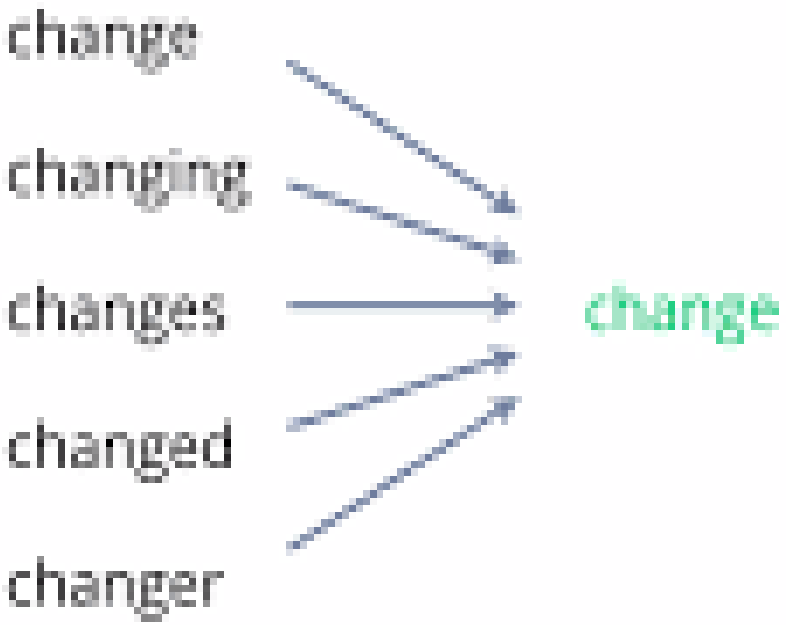
LEMMATIZATION



PART OF SPEECH (POS)

Word	POS Tag	Word Class
Ice	NN	noun (singular)
skates	NNS	noun (plural)
are	VBP	verb (plural)
boots	NNS	noun (plural)
with	IN	preposition
blades	NNS	noun (plural)
attached	VBN	verb (past participle)
to	TO	"to"
it	PRP	personal pronoun

Stemming vs Lemmatization



Practical

- How to extract text from pdf, text file using python
- How vector representation works with python
- How NLTK and spaCy works through with python