Text analytics & Natural Language Processing

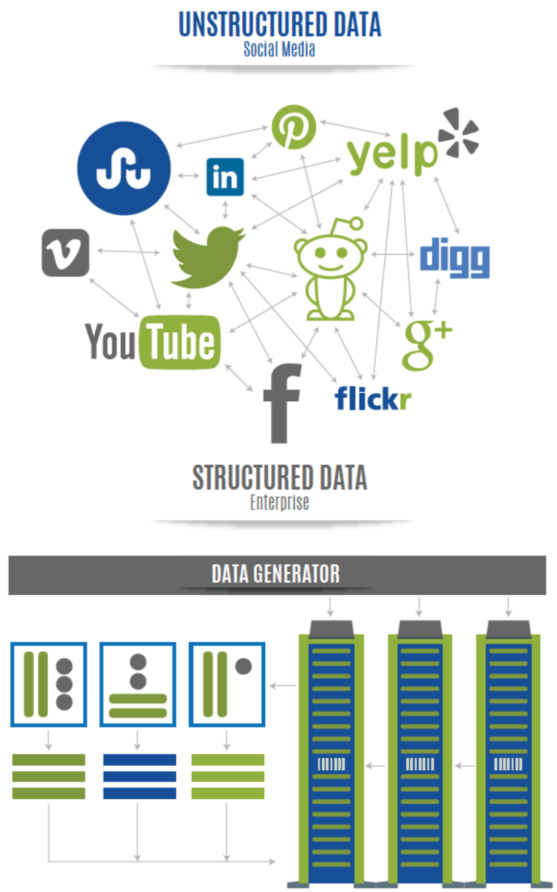
Kecerdasan Buatan

Teknik Informatika - Politeknik Elektronika Negeri Surabaya

Topics

- Text Analytics and NLP
- Compare Text Analytics, NLP and Text Mining
  - Text Analysis Operations using NLTK
  - Tokenization
  - Stopwords
  - Lexicon Normalization such as Stemming and Lemmatization
  - POS Tagging
- Sentiment Analysis
- Text Classification
- Performing Sentiment Analysis using Text Classification

Introduction (1)



- In today's area of internet and online services, data is generating at incredible speed and amount.
- Generally, data analyst, engineer, and scientists are handling relational or tabular data.
- These tabular data columns have either numerical or categorical data.
- Generated data has a variety of structures such as text, image, audio, and video.
- Online activities such as articles, website text, blog posts, social media posts are generating unstructured textual data.

Image source: smartdatacollective.com, A Quick Guide to Structured and Unstructured Data

Introduction (2)

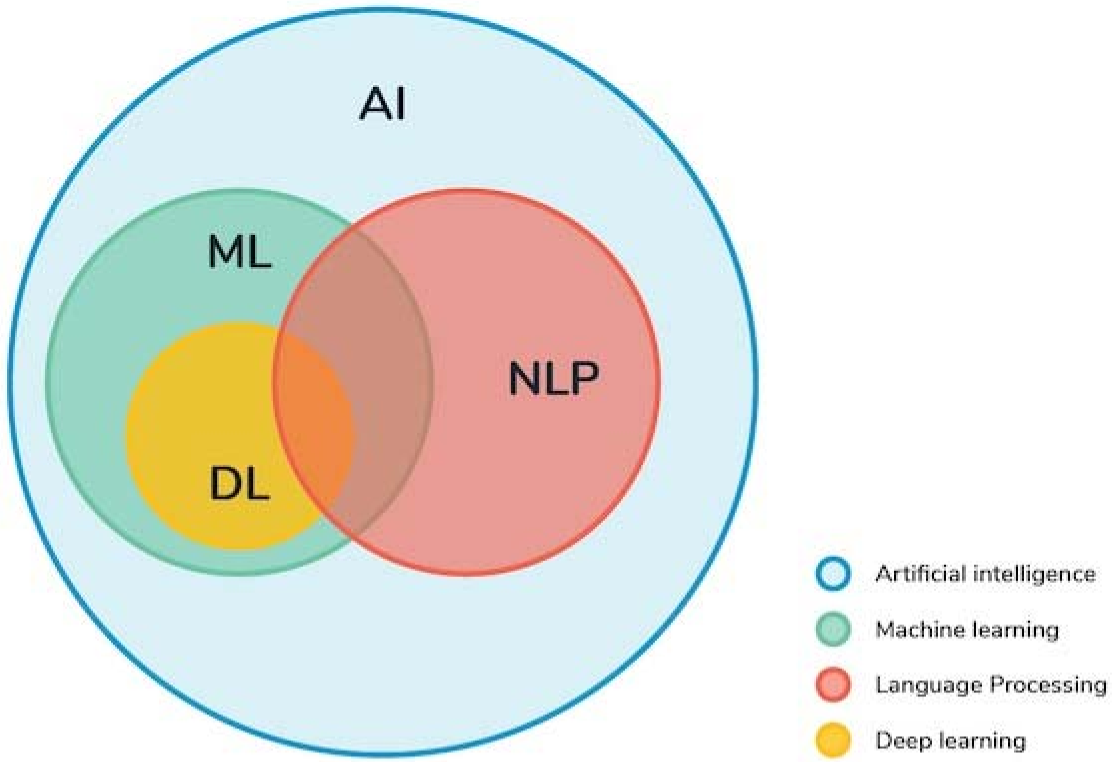
- Text communication is one of the most popular forms of day to day conversion.
- Text communication in our daily routine:
  - chat
  - message
  - tweet
  - share status
  - email
  - write blogs
  - share opinion and feedback.

- Generates text in a significant amount.
- It is unstructured in nature.

Introduction (3)

- In this area of the online marketplace and social media, it is essential to analyze vast quantities of data, to understand peoples opinion.
- Corporate and business need to analyze textual data to understand customer activities, opinion, and feedback to successfully derive their business.
- To compete with big textual data, text analytics is evolving at a faster rate than ever before.

Introduction (4)



Source: https://www.geekword.net/text-mining-nlp-natural-language-processing

Text Analytics applications in today's online world

- By analyzing
  - Tweets on Twitter, we can find trending news and peoples reaction on a particular event.
  - Amazon can understand user feedback or review on the specific product.
  - BookMyShow can discover people's opinion about the movie.
  - Youtube can also analyze and understand peoples viewpoints on a video.

Text Analytics & Text Mining

- Nama lain dari text analytics adalah text mining.
- Text Analytics adalah metode yang digunakan untuk memperoleh data structured berkualitas tinggi dari kumpulan text unstructured.

Text Mining

- Text mining merupakan penerapan konsep dan teknik data mining untuk mencari pola dalam teks
- Teks Mining : Proses penganalisisan teks guna menyarikan informasi yang bermanfaat untuk tujuan tertentu.
- Kegiatan text mining, finding:
  - frequency counts of word
  - length of the sentence
  - presence/absence of specific words
- Natural language processing (NLP) is one of the components of text mining.

Text Mining Vs. Data Mining

	Data Mining	Text Mining
Data Object	Numerical & categorical data	Textual data
Data structure	Structured	Unstructured &semi-structured
Data representation	Straightforward	Complex
Space dimension	< tens of thousands	> tens of thousands
Methods	Data analysis, machine learning, Data mining, information	Statistic, neural networks retrieval, NLP, ...
Maturity	Broad implementation since1994	Broad implementation starting 2000
Market	10^5 analysts at large and mid size companies	10^8 analysts corporate workers and individual users

Source: Pankaj Thakur, Depertment of CSE, NITTTR Chandigarrh

Sumber data yang dapat dianalisa oleh text analytics

Data-data yang berada di dalam:

- media sosial seperti Facebook, LinkedIn, Twitter, Instagram
- isi email
- artikel berita
- online discussion forum
- review website (TripAdvisor)
- PDF documents
- online forms

Natural Language Processing (1)

- Natural Language Processing (NLP) is the technology used to aid computers to understand the human's natural language.
- It's not an easy task teaching machines to understand how we communicate.
- NLP enables the computer to interact with humans in a natural manner.
- It helps the computer to understand the human language and derive meaning from it.

Natural Language Processing (2)

- Bahasa alami adalah bahasa yang secara umum digunakan oleh manusia dalam berkomunikasi satu sama lain.
- Bahasa yang diterima oleh komputer butuh diproses dan dipahami terlebih dahulu supaya maksud dari user bisa dipahami dengan baik oleh komputer.

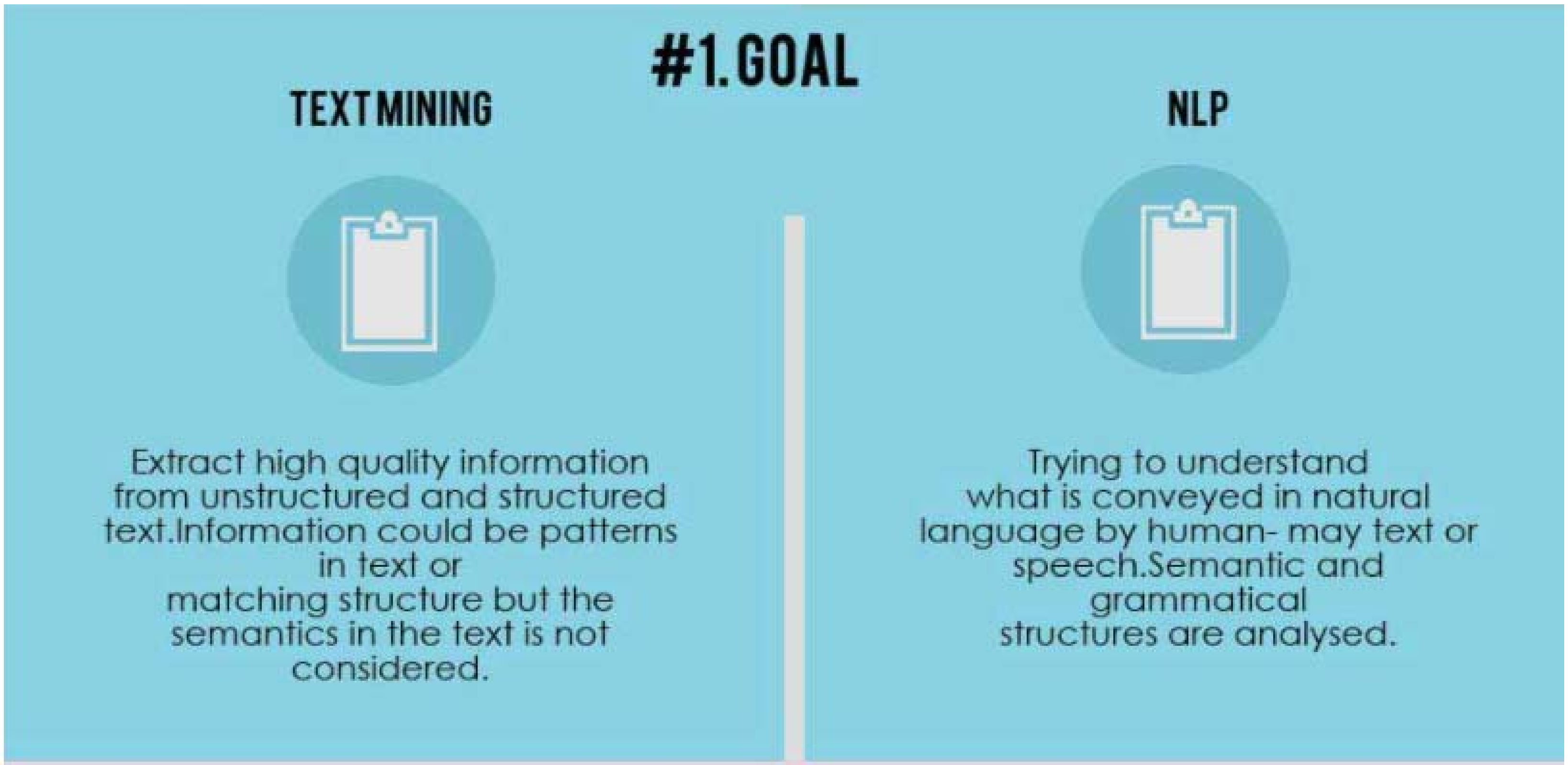
Natural Language Processing (2)

- NLP is applicable in several problematic from:
  - speech recognition
  - language translation
  - classifying documents
  - information extraction
- Analyzing movie review is one of the classic examples to demonstrate a simple NLP Bag-of-words model, on movie reviews.

What is NLP used for?

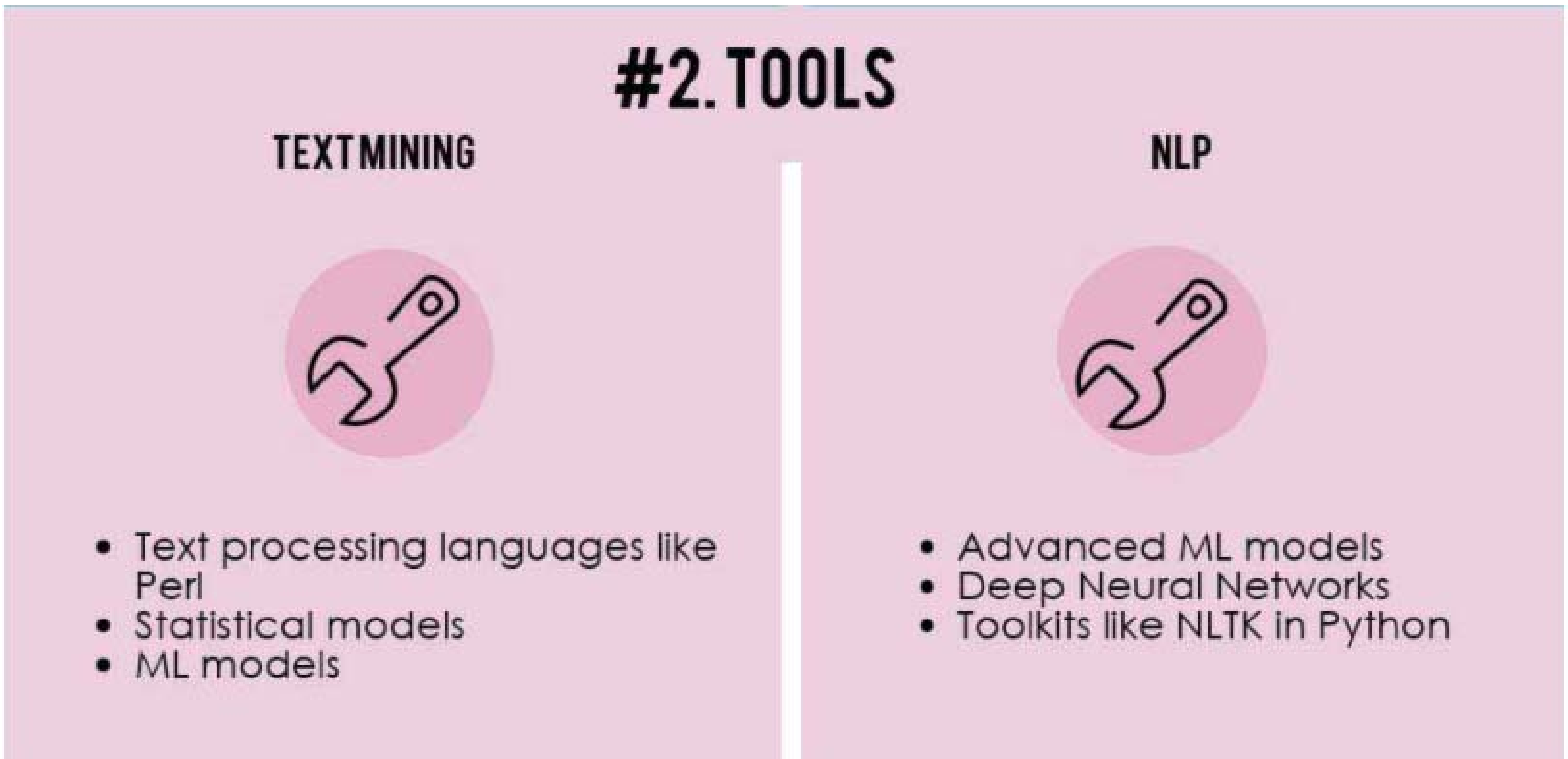
- Language translation applications such as Google Translate
- Word Processors such as Microsoft Word and Grammarly that employ NLP to check grammatical accuracy of texts.
- Interactive Voice Response (IVR) applications used in call centers to respond to certain users' requests.
- Personal assistant applications such as OK Google, Siri, Cortana, and Alexa.

Text Mining vs. NLP



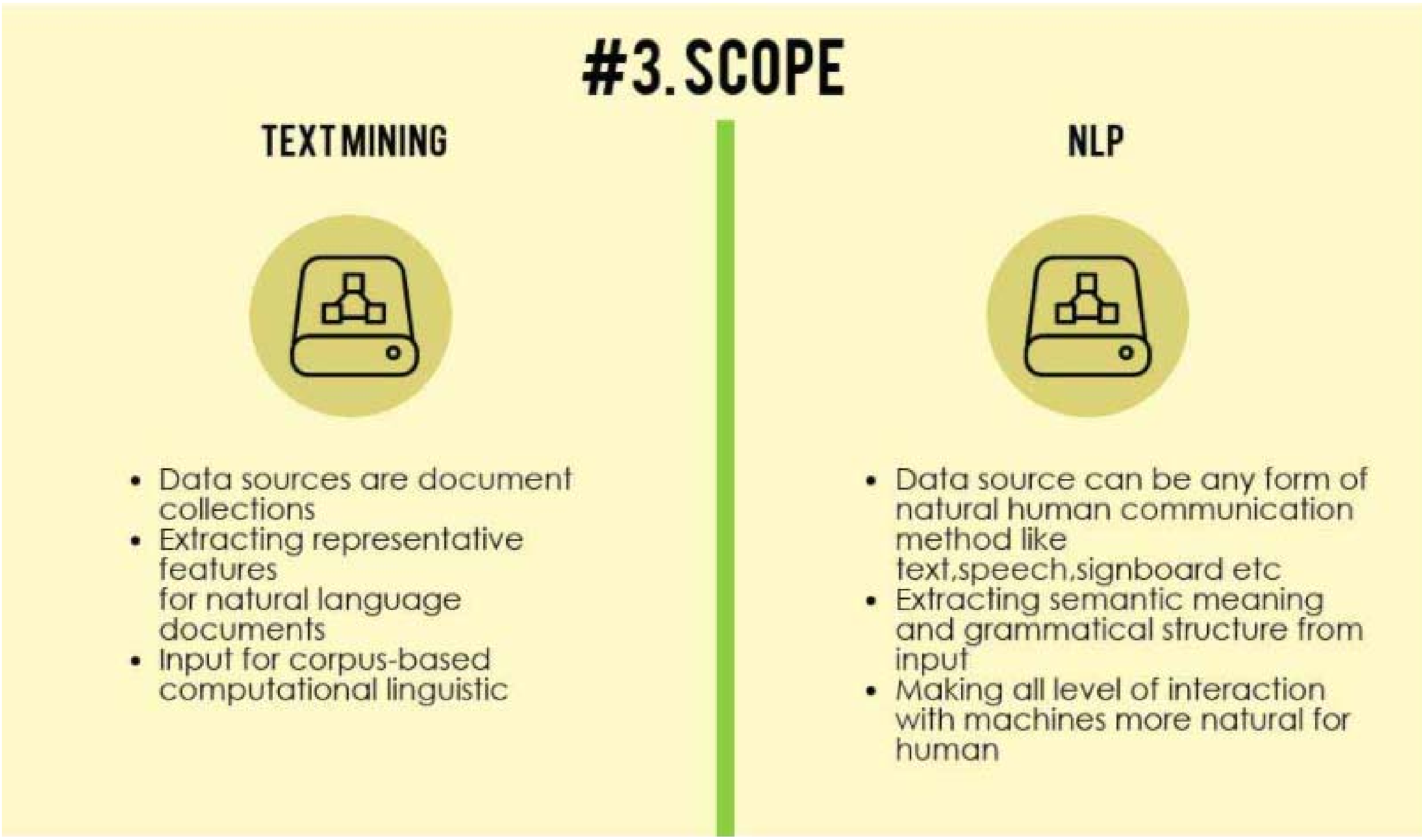
https://www.educba.com/important-text-mining-vs-natural-language-processing/

Text Mining vs. NLP



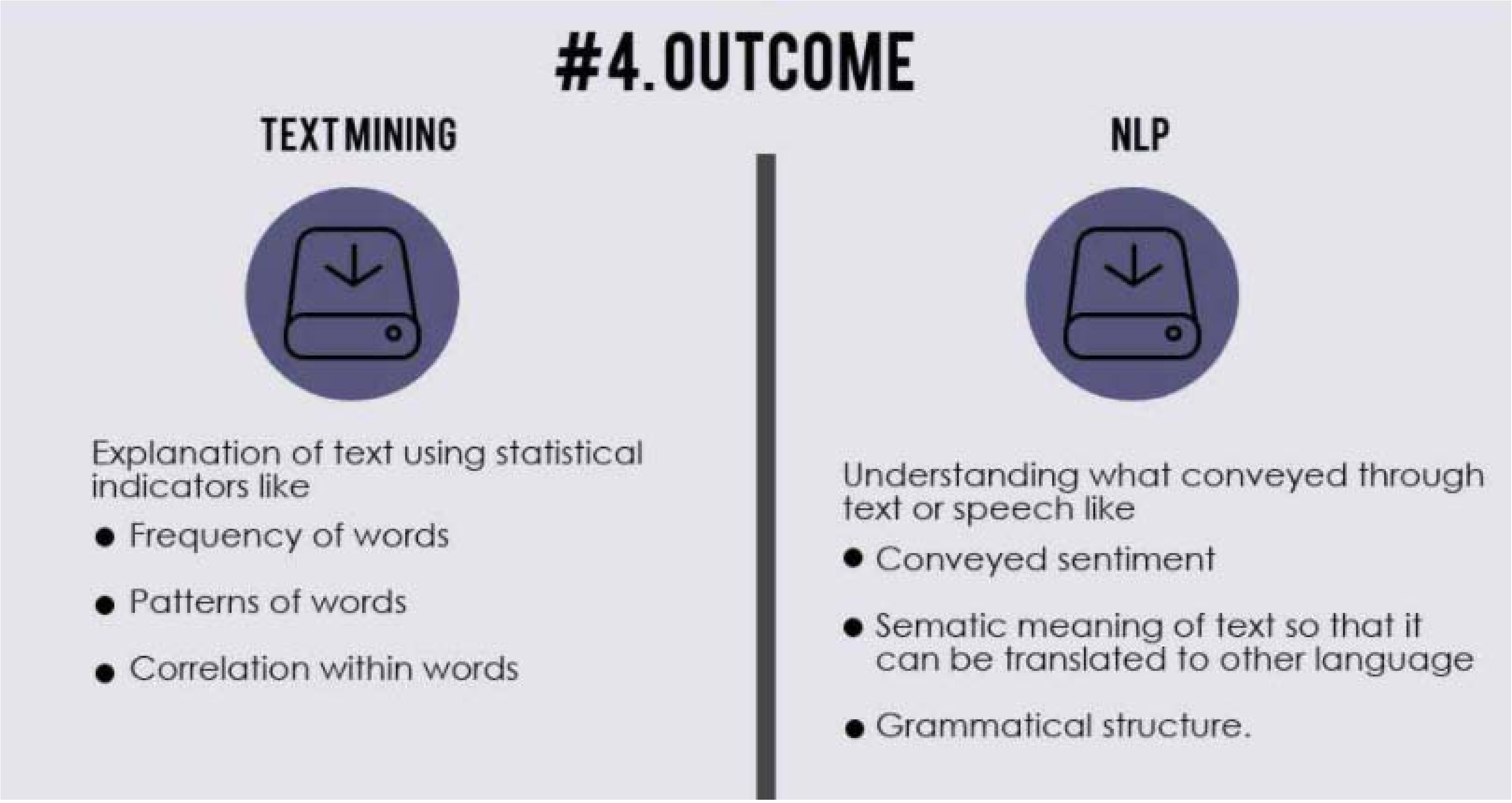
https://www.educba.com/important-text-mining-vs-natural-language-processing/

Text Mining vs. NLP



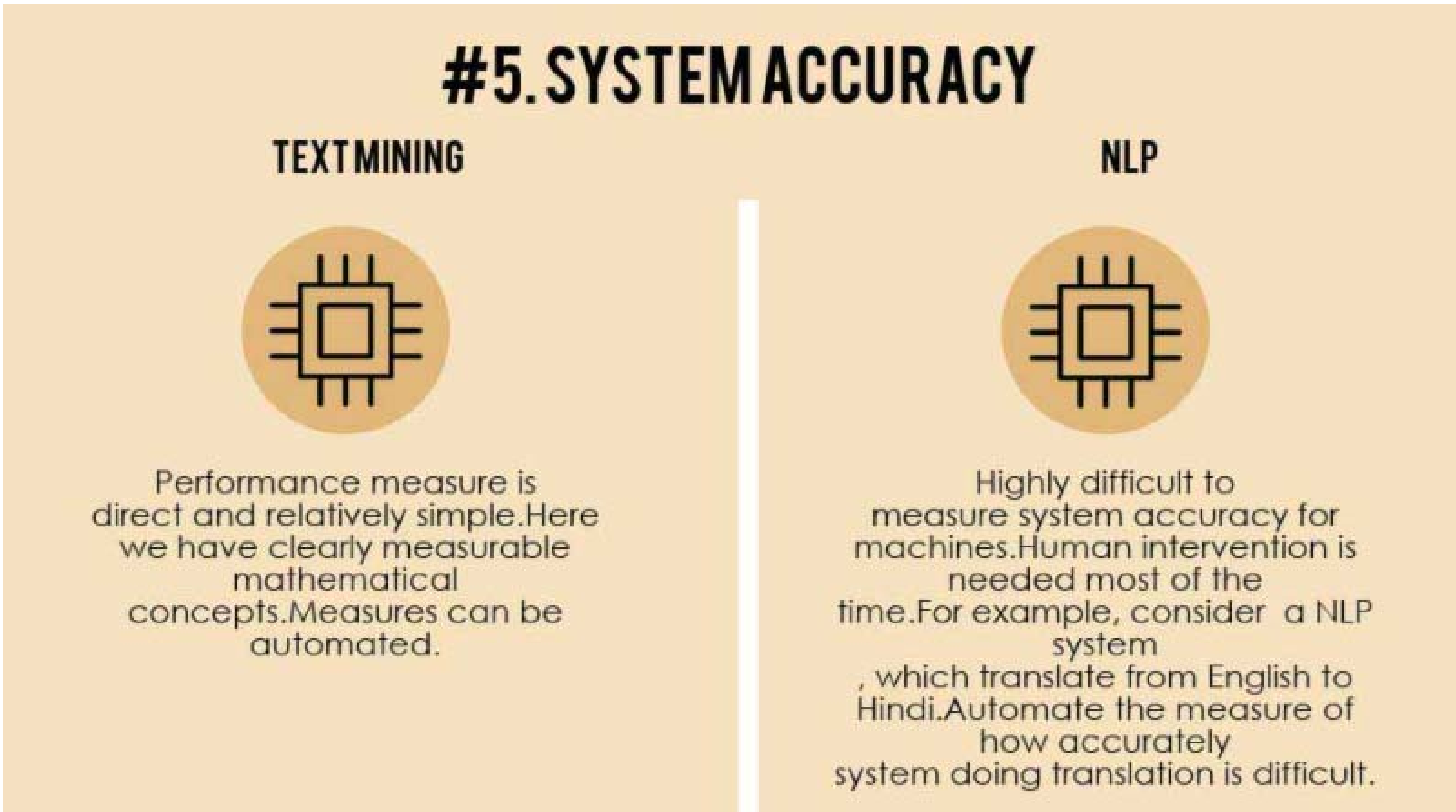
https://www.educba.com/important-text-mining-vs-natural-language-processing/

Text Mining vs. NLP



https://www.educba.com/important-text-mining-vs-natural-language-processing/

Text Mining vs. NLP



https://www.educba.com/important-text-mining-vs-natural-language-processing/

Text Mining	Natural Language Processing
Aim of text mining is to extract useful insights from structured & un-structured text.	Aim of NLP is to understand what is conveyed in speech.
Text Mining can be done using text processing languages like Perl, statistical models, etc.	NLP can be achieved using advanced machine learning models, deep neural networks, etc.
Outcome: • Frequency of words • Patterns • Correlations	Outcome: • Semantic meaning of text • Sentimental analysis • Grammatical structure

Source: https://medium.com/@emmanuel.balraj_57030/nlp-and-text-mining-374f7796a773

NLP Tools (1)

Parsing Tools – English

- Berkeley Parser:
  - http://tomato.banatao.berkeley.edu:8080/parser/parser.html
- Stanford CoreNLP
  - http://nlp.stanford.edu:8080/corenlp/
- Stanford Parser
  - Support: English, Simplified Chinese, Arbic, French, Spanish
  - http://nlp.stanford.edu:8080/parser/index.jsp

NLP Tools (2)

Parsing Tools - Chinese

- Stanford Parser
  - Support: English, Simplified Chinese, Arabic, French, Spanish
  - http://nlp.stanford.edu:8080/parser/index.jsp
- 語言雲
  - https://www.ltp-cloud.com/intro/
- CKIP (Traditional Chinese)
  - http://ckipsvr.iis.sinica.edu.tw/

NLP Tools (3)

More Tools

- NLTK (python): tokenize, tag, NE extraction, show parsing tree
  - Porter stemmer
  - n-grams
- spyCy: industrial-strength NLP in python
- Apache OpenNLP;
- Stanford NLP suite;
- Gate NLP library

Why is NLP difficult?

- It's the nature of the human language that makes NLP difficult.
- The rules that dictate the passing of information.
  - Some of these rules can be high-leveled and abstract; for example, when someone uses a sarcastic remark to pass information.
  - Some of these rules can be low-levelled; for example, using the character "s" to signify the plurality of items.
- Comprehensively understanding the human language requires understanding both the words and how the concepts are connected to deliver the intended message.
- The ambiguity and imprecise characteristics of the natural languages are what make NLP difficult for machines to implement.

Text Analytics Operations using NLTK

- NLTK is a powerful Python package that provides a set of diverse natural languages algorithms.
- It is free, opensource, easy to use, large community, and well documented.
- NLTK consists of the most common algorithms such as:
  - Tokenizing
  - part-of-speech tagging
  - Stemming
  - sentiment analysis
  - topic segmentation
  - named entity recognition
- NLTK helps the computer to analysis, preprocess, and understand the written text.

Instal NLTK

```
!pip install nltk
```

Requirement already satisfied: nltk in /home/northout/anaconda2/lib/python2.7/site-packages
Requirement already satisfied: six in /home/northout/anaconda2/lib/python2.7/site-packages (from nltk)
[33mYou are using pip version 9.0.1, however version 10.0.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.[0m

```
#Loading NLTK
import nltk
```

Tokenization

- Tokenization is the first step in text analytics.
- Tokenization is the process of breaking down a text paragraph into smaller chunks such as:
  - Words
  - sentence.
- Token is a single entity that is building blocks for sentence or paragraph.

Labs 1: Tokenization
Sentence Tokenization

- Sentence tokenizer breaks text paragraph into sentences.
- Here, the given text is tokenized into sentences.

```
from nltk.tokenize import sent_tokenize
nltk.download('punkt')

text="""Hello Mr. Smith, how are you doing today? The weather is great,
 and city is awesome.
The sky is pinkish-blue. You shouldn't eat cardboard"""
tokenized_text = sent_tokenize(text)
print(tokenized_text)
```

```
['Hello Mr. Smith, how are you doing today?', 'The weather is great, and city is awesome.', '
```

Tokenization: Word Tokenization

- Word tokenizer breaks text paragraph into words.

```
from nltk.tokenize import word_tokenize

tokenized_word=word_tokenize(text)
print(tokenized_word)
```

```
['Hello', 'Mr.', 'Smith', ',', 'how', 'are', 'you', 'doing', 'today', '?', 'The', 'weather',
```

Frequency Distribution (1)

```
from nltk.probability import FreqDist
fdist = FreqDist(tokenized_word)
print(fdist)
```

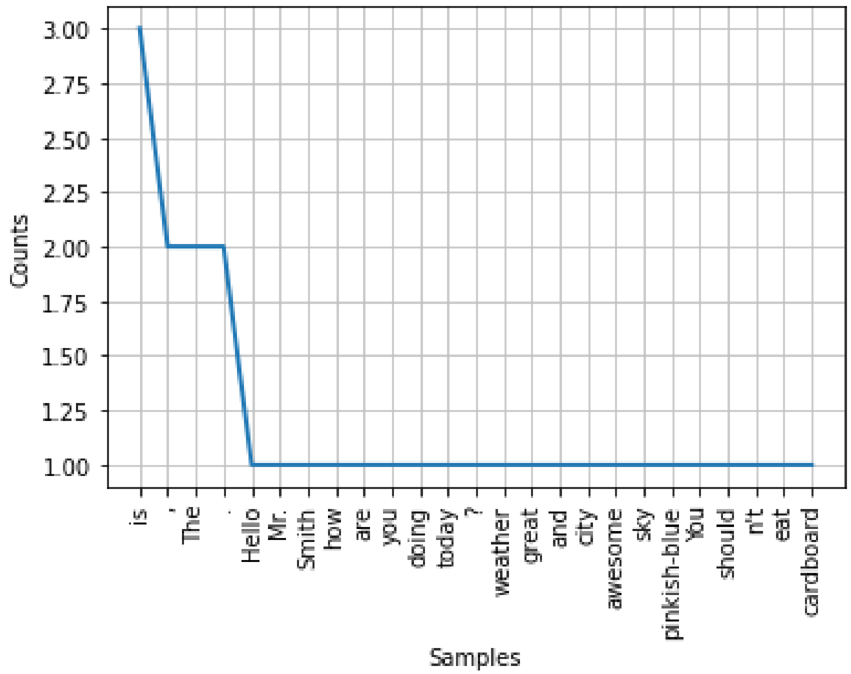
```
<FreqDist with 25 samples and 30 outcomes>
```

```
fdist.most_common(2)
```

```
[('is', 3), (',', 2)]
```

```
# Frequency Distribution Plot
import matplotlib.pyplot as plt

fdist.plot(30,cumulative=False)
plt.show()
```



Stopwords (1)

- Stopwords considered as noise in the text.
- Text may contain stop words such as:
  - Is
  - am
  - are
  - this
  - a
  - an
  - the, etc.
- In NLTK for removing stopwords, you need to create a list of stopwords and filter out your list of tokens from these words.

Stopwords (2)

```
from nltk.corpus import stopwords
nltk.download('stopwords')

stop_words=set(stopwords.words("english"))
print(stop_words)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
{'itself', 'so', 'until', 'now', "it's", "you're", 'those', 'o', 'hers', 'our', 'whom', 've',
```

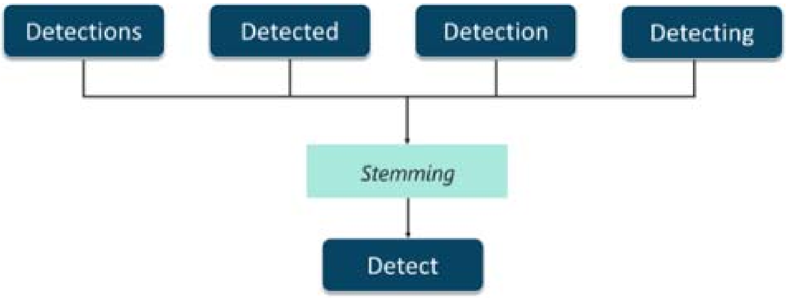
Removing Stopwords

```
# Removing Stopwords
filtered_sent=[]
for w in tokenized_word:
    if w not in stop_words:
        filtered_sent.append(w)
print("Tokenized Sentence:",tokenized_word)
print("Filterd Sentence:",filtered_sent)
```

```
Tokenized Sentence: ['Hello', 'Mr.', 'Smith', ',', 'how', 'are', 'you', 'doing', 'today', '?', 'The',
Filterd Sentence: ['Hello', 'Mr.', 'Smith', ',', 'today', '?', 'The', 'weather', 'great', ',', 'city',
```

Lexicon Normalization: Stemming

- Lexicon normalization considers another type of noise in the text.
- It reduces derivationally related forms of a word to a common root word.
- Stemming is a process of linguistic normalization, which reduces words to their word root word or chops off the derivational affixes.
- For example:
  - connection, connected, connecting word reduce to a common word "connect".



Lexicon Normalization: Stemming

```python
# Lexicon Normalization: Stemming
# Stemming
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize

ps = PorterStemmer()

stemmed_words=[]
for w in filtered_sent:
    stemmed_words.append(ps.stem(w))

print("Filtered Sentence:",filtered_sent)
print("Stemmed Sentence:",stemmed_words)
```

Lexicon Normalization: Lemmatization

- Lemmatization reduces words to their base word, which is linguistically correct lemmas.
- It transforms root word with the use of vocabulary and morphological analysis.
- Lemmatization is usually more sophisticated than stemming.
- Stemmer works on an individual word without knowledge of the context.
- For example:

  The word "better" has "good" as its lemma.

  The word “went" has “go" as its lemma
- This thing will miss by stemming because it requires a dictionary look-up.

Lemmatization

```
# Lemmatization
#Lexicon Normalization
#performing stemming and Lemmatization

from nltk.stem.wordnet import WordNetLemmatizer
nltk.download('wordnet')
lem = WordNetLemmatizer()

from nltk.stem.porter import PorterStemmer
stem = PorterStemmer()

word = "flying"
print("Lemmatized Word:",lem.lemmatize(word,"v"))
print("Stemmed Word:",stem.stem(word))
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
Lemmatized Word: fly
Stemmed Word: fli
```

POS Tagging (1)

- The primary target of Part-of-Speech(POS) tagging is to identify the grammatical group of a given word.
- POS Tagging looks for relationships within the sentence and assigns a corresponding tag to the word.
- Based on the context, whether it is a:
  - NOUN
  - PRONOUN
  - ADJECTIVE
  - VERB
  - ADVERBS

POS Tagging (2)

```
# POS Tagging
sent = "Albert Einstein was born in Ulm, Germany in 1879.“

Tokens = nltk.word_tokenize(sent)
print(tokens)
```

```
['Albert', 'Einstein', 'was', 'born', 'in', 'Ulm', ',', 'Germany', 'in', '1879', '.']
```

POS Tagging (3)

```
nltk.download('averaged_perceptron_tagger')
nltk.pos_tag(tokens)
```

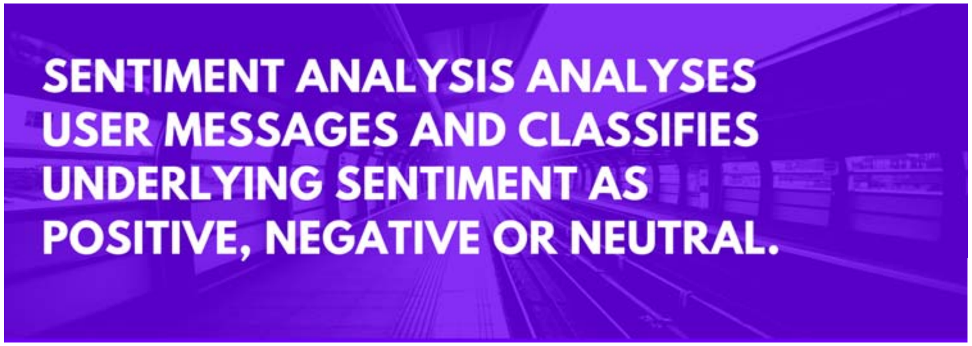
```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[('Albert', 'NNP'),
 ('Einstein', 'NNP'),
 ('was', 'VBD'),
 ('born', 'VBN'),
 ('in', 'IN'),
 ('Ulm', 'NNP'),
 (',', ','),
 ('Germany', 'NNP'),
 ('in', 'IN'),
 ('1879', 'CD'),
 ('.', '.')]
```

Sentiment Analysis (1)

- Companies want to understand:
  - What went wrong with their latest products?
  - What users and the general public think about the latest feature?
- Sentiment analysis can be used to quantify such information with reasonable accuracy.
- Quantifying users content, idea, belief, and opinion.
- Sentiment analysis helps in understanding people in a better and more accurate way.

Sentiment Analysis (2)

- Human communication just not limited to words, it is more than words.
- Sentiments are combination of:
  - Words
  - Tone
  - Writing style



Sentiment Analysis (3)

There are mainly two approaches for performing sentiment analysis.

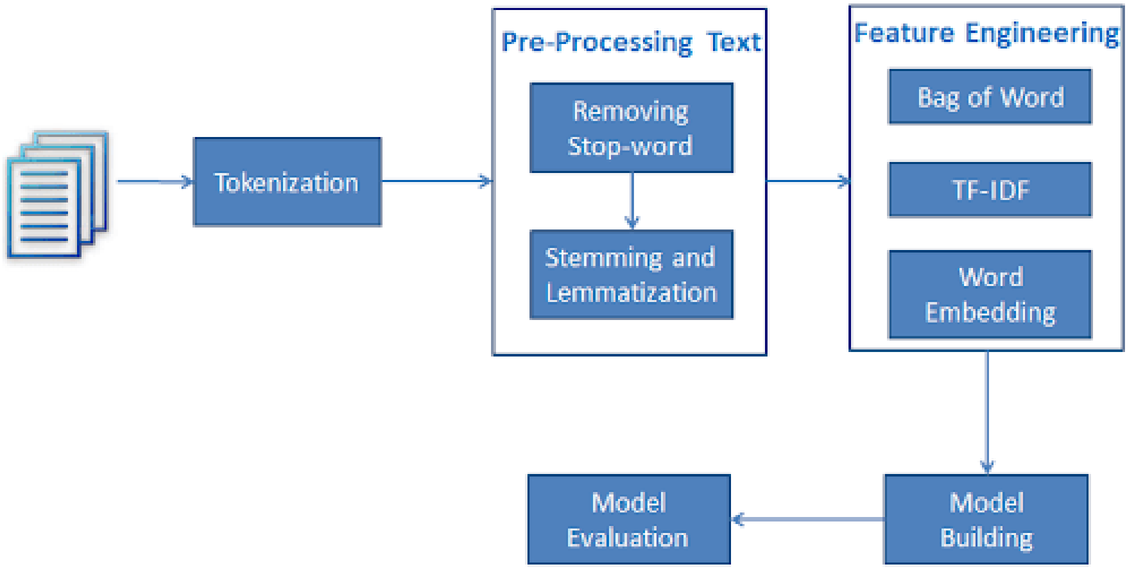
- Lexicon-based:

  Count number of positive and negative words in given text and the larger count will be the sentiment of text.
- Machine learning based:

  Develop a classification model, which is trained using the pre-labeled dataset of positive, negative, and neutral.

Performing Sentiment Analysis using Text Classification

- Text classification is one of the important tasks of text mining.
- It is a supervised approach.
- Identifying category or class of given text such as a blog, book, web page, news articles, and tweets.



Labs: Performing Sentiment Analysis using Text Classification

Loading Data

- We have learned data pre-processing using NLTK.
- Now, we will learn Text Classification.
- We will perform Multi-Nominal Naive Bayes Classification using scikit-learn.

- In the model the building part, we can use the "Sentiment Analysis of Movie, Reviews" dataset available on Kaggle.
- The dataset is a tab-separated file.
- Dataset has four columns PhraseId, SentenceId, Phrase, and Sentiment.

Labs: Performing Sentiment Analysis using Text Classification

- This data has 5 sentiment labels:
  - 0 - negative
  - 1 - somewhat negative
  - 2 - neutral
  - 3 - somewhat positive
  - 4 - positive

- The dataset is available on Kaggle.
- We can download it from the following link: https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/data

Labs: Performing Sentiment Analysis using Text Classification

```
# Import pandas
import pandas as pd

data=pd.read_csv('train.tsv', sep='\t')
data.head()
```

	PhraseId	SentenceId	Phrase	Sentiment
0	1	1	A series of escapades demonstrating the adage ...	1.0
1	2	1	A series of escapades demonstrating the adage ...	2.0
2	3	1	A series	2.0
3	4	1	A	2.0
4	5	1	series	2.0

Labs: Performing Sentiment Analysis using Text Classification

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22143 entries, 0 to 22142
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   PhraseId    22143 non-null  int64  
 1   SentenceId  22143 non-null  int64  
 2   Phrase      22143 non-null  object 
 3   Sentiment   22142 non-null  float64
dtypes: float64(1), int64(2), object(1)
memory usage: 692.1+ KB
```

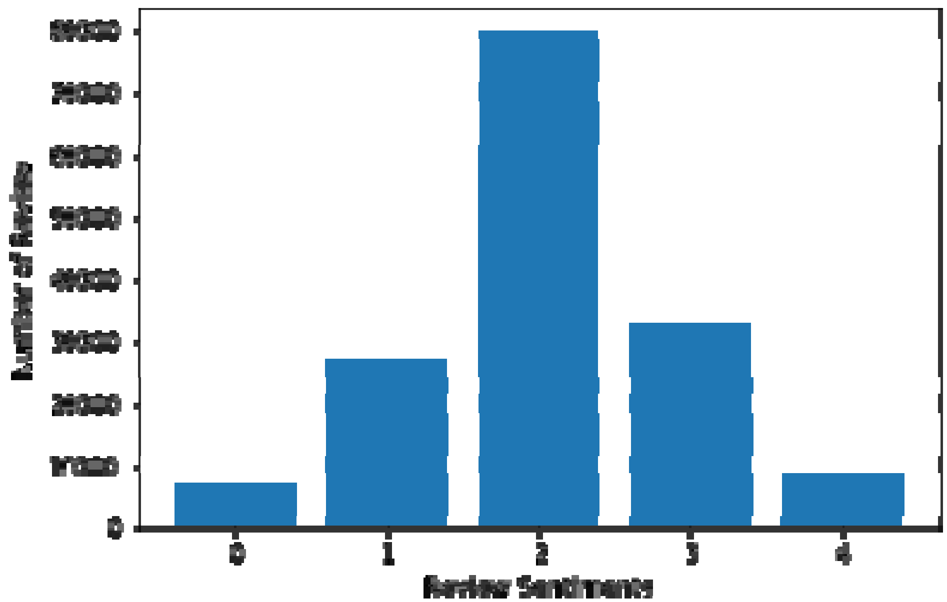
```
data.Sentiment.value_counts()
```

```
2.0    12287
3.0     4451
1.0     3471
4.0     1159
0.0      774
Name: Sentiment, dtype: int64
```

Labs: Performing Sentiment Analysis using Text Classification

```
import matplotlib.pyplot as plt

Sentiment_count=data.groupby('Sentiment').count()
plt.bar(Sentiment_count.index.values, Sentiment_count['Phrase'])
plt.xlabel('Review Sentiments')
plt.ylabel('Number of Review')
plt.show()
```



Labs: Performing Sentiment Analysis using Text Classification Feature Generation using Bag of Words (BoW)

- In the Text Classification Problem, we have a set of texts and their respective labels.
- But we directly can't use text for our model.
- We need to convert these text into some numbers or vectors of numbers.
- Bag-of-words model (BoW) is the simplest way of extracting features from the text.
- BoW converts text into the matrix of occurrence of words within a document.
- This model concerns about whether given words occurred or not in the document.

Labs: Performing Sentiment Analysis using Text Classification
Feature Generation using Bag of Words (BoW)

- Example: There are three documents:
  - Doc 1: I love dogs.
  - Doc 2: I hate dogs and knitting.
  - Doc 3: Knitting is my hobby and passion.
- Now, we can create a matrix of document and words by counting the occurrence of words in the given document.
- This matrix is known as Document-Term Matrix (DTM).
- This matrix is using a single word. It can be a combination of two or more words, which is called a bigram or trigram model and the general approach is called the n-gram model.

	I	love	dogs	hate	and	knitting	is	my	hobby	passion
Doc 1	1	1	1							
Doc 2	1		1	1	1	1				
Doc 3					1	1	1	2	1	1

Labs: Performing Sentiment Analysis using Text Classification Feature Generation using Bag of Words (BoW)

Generate document term matrix by using scikit-learn's CountVectorizer.

```
# Feature Generation using Bag of Words
from sklearn.feature_extraction.text import CountVectorizer
from nltk.tokenize import RegexpTokenizer

#tokenizer to remove unwanted elements from out data like symbols and numbe
rs
token = RegexpTokenizer(r'[a-zA-Z0-9]+')
cv = CountVectorizer(lowercase=True,stop_words='english',ngram_range = (1,1
),tokenizer = token.tokenize)
text_counts= cv.fit_transform(data['Phrase'])
```

Labs: Performing Sentiment Analysis using Text Classification Feature Generation using Bag of Words (BoW)

Split train and test set

- To understand model performance, dividing the dataset into a training set and a test set is a good strategy.
- Let's split dataset by using function train_test_split().
- We need to pass basically 3 parameters features, target, and test_set size.
- Additionally, we can use random_state to select records randomly.

```
# Split train and test set

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    text_counts, data['Sentiment'], test_size=0.3, random_state=1)
```

Labs: Performing Sentiment Analysis using Text Classification

Model Building and Evaluation

- Let's build the Text Classification Model using TF-IDF.
- First, import the MultinomialNB module and create a Multinomial Naive Bayes classifier object using MultinomialNB() function.
- Then, fit your model on a train set using fit() and perform prediction on the test set using predict().

Labs: Performing Sentiment Analysis using Text Classification

Model Building and Evaluation

```
# Model Building and Evaluation
from sklearn.naive_bayes import MultinomialNB
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
import numpy as np

# Model Generation Using Multinomial Naive Bayes
X_train = np.nan_to_num(X_train)
y_train = np.nan_to_num(y_train)
clf = MultinomialNB().fit(X_train, y_train)
predicted= clf.predict(X_test)
y_test = np.nan_to_num(y_test)
print("MultinomialNB Accuracy:",metrics.accuracy_score(y_test, predicted))
```

MultinomialNB Accuracy: 0.6137287370164083

Labs: Performing Sentiment Analysis using Text Classification

Feature Generation using TF-IDF

- In Term Frequency (TF), you just count the number of words occurred in each document.
- The main issue with this Term Frequency is that it will give more weight to longer documents.
- Term frequency is basically the output of the BoW model.
- IDF (Inverse Document Frequency) measures the amount of information a given word provides across the document.
- IDF is the logarithmically scaled inverse ratio of the number of documents that contain the word and the total number of documents.

$$\mathrm{idf}(\mathrm{W}) = \log \frac{\#(\text{documents})}{\#(\text{documents containing word W})}$$

Labs: Performing Sentiment Analysis using Text Classification
Feature Generation using TF-IDF

- TF-IDF (Term Frequency-Inverse Document Frequency) normalizes the document term matrix.
- It is the product of TF and IDF.
- Word with high tf-idf in a document, it is most of the times occurred in given documents and must be absent in the other documents.
- So the words must be a signature word.

	I	love	dogs	hate	and	knitting	is	my	hobby	passion
Doc 1	0.18	**0.48**	0.18							
Doc 2	0.18		0.18	**0.48**	0.18	0.18				
Doc 3					0.18	0.18	**0.48**	**0.95**	**0.48**	**0.48**

Labs: Performing Sentiment Analysis using Text Classification

Feature Generation using TF-IDF

```
from sklearn.feature_extraction.text import TfidfVectorizer
tf=TfidfVectorizer()
text_tf= tf.fit_transform(data['Phrase'])
```

Labs: Performing Sentiment Analysis using Text Classification

Split train and test set (TF-IDF)

- Let's split dataset by using function train_test_split().
- We need to pass:
    - basically 3 parameters features
    - Target
    - test_set size
- Additionally, you can use random_state to select records randomly.

```
# Split train and test set (TF-IDF)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    text_tf, data['Sentiment'], test_size=0.3, random_state=123)
```

Labs: Performing Sentiment Analysis using Text Classification

Model Building and Evaluation (TF-IDF)

- Let's build the Text Classification Model using TF-IDF.
- First, import the MultinomialNB module and create the Multinomial Naive Bayes classifier object using MultinomialNB() function.
- Then, fit your model on a train set using fit() and perform prediction on the test set using predict().

```
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
# Model Generation Using Multinomial Naive Bayes
X_train = np.nan_to_num(X_train)
y_train = np.nan_to_num(y_train)
clf = MultinomialNB().fit(X_train, y_train)
predicted= clf.predict(X_test)
print("MultinomialNB Accuracy:",metrics.accuracy_score(y_test, predicted))
```

```
MultinomialNB Accuracy: 0.6018365196447388
```

Conclusion (1)

In this topics, we have learned:

- Text Analytics
- NLP and text mining
- Basics of text analytics operations using NLTK such as:
  - Tokenization
  - Normalization
  - Stemming
  - Lemmatization
  - POS tagging
- Sentiment analysis and text classification using scikit-learn.

Conclusion (2)

- Both Text Mining vs Natural Language Processing trying to extract information from unstructured data.
- Text mining is concentrated on text documents and mostly depends on a statistical and probabilistic model to derive a representation of documents.
- NLP trying to get semantic meaning from all means of human natural communication like text, speech or even an image.
- NLP has the potential to revolutionize the way humans interact with machines

References

- Avinash Navlani, Text Analytics for Beginners using NLTK, December 14th, 2019. https://www.datacamp.com/community/tutorials/text-analytics-beginners-nltk
- Michelle Chen, A Guide: Text Analysis, Text Analytics & Text Mining, 21 October 2020. https://towardsdatascience.com/a-guide-text-analysis-text-analytics-text-mining-f62df7b78747
- Michael J. Garbade, A Simple Introduction to Natural Language Processing, Oct 15, 2018, https://becominghuman.ai/a-simple-introduction-to-natural-language-processing-ea66a1747b32
- Priya Pedamkar, Text Mining vs Natural Language Processing, https://www.educba.com/important-text-mining-vs-natural-language-processing/