Sane's STATS Academy of Statistics

Text Mining

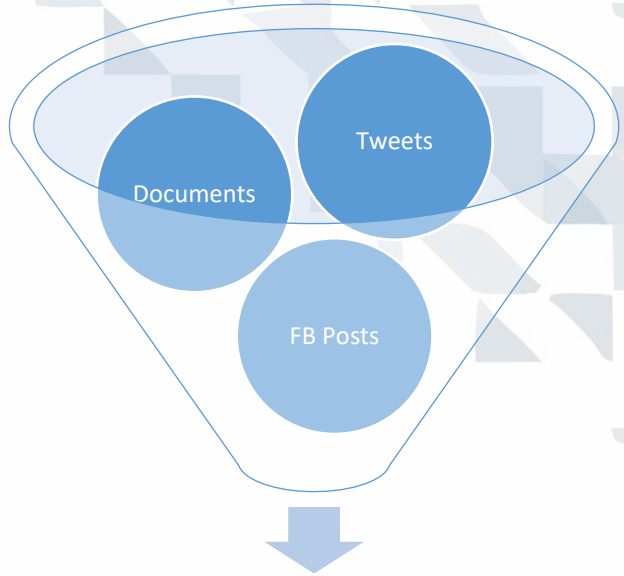
An Introduction

What is Text Mining?

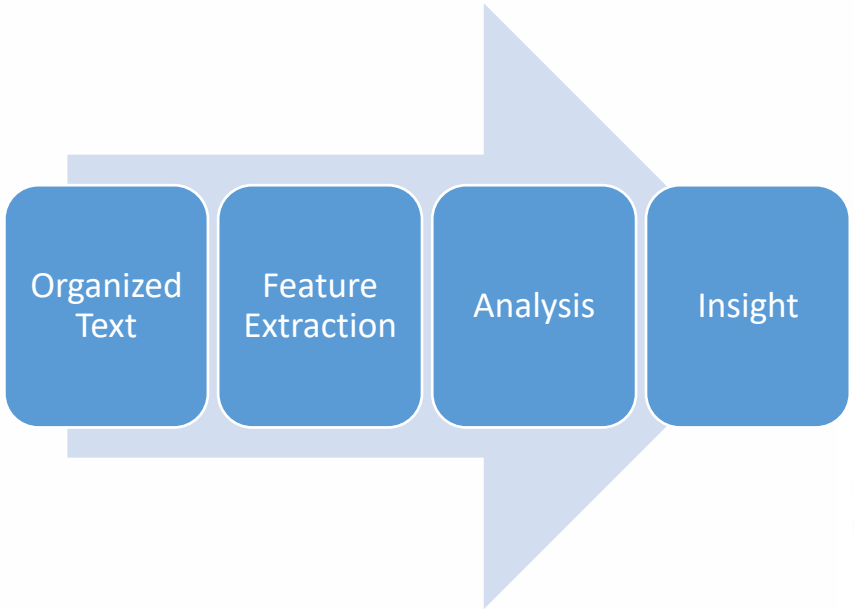
- Finding actionable insights from text
- Text Mining helps you to dig out meaningful information from a bunch of tweets / documents

Text Mining Workflow

1. Define the problem and specify what you want
2. Identify the text to be collected; e.g. Documents, emails, tweets etc.
3. Organize the text; e.g. by author, chronologically
4. Feature Extraction; e.g. calculating sentiments, extracting word tokens
5. Analysis
6. Reach an insight / recommendation



Organize the text



Sane's STATS Academy of Statistics

What is Corpus?

- A collection of written texts, especially the entire works of a particular author or a body of writing on a particular subject

Approaches to Text Mining

- Bag of Words
- Word Embedding

Sane's STATS Academy of Statistics

Bag of Words

Cleaning the text

- Before applying any pre-processing, we consider cleaning the text
- Text can be cleaned by stemming, lemmatising, removing stop words etc.

Stemming the Words

- **Stemming** is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form—generally a written word form.
- In package nltk.PorterStemmer, the method stem() uses Porter's Stemming Algorithm and heuristically completes the stemmed words.

e.g.

```
In [7]: from nltk import PorterStemmer

In [8]: PorterStemmer().stem('complications')
Out[8]: 'complic'

In [9]: PorterStemmer().stem('complicated')
Out[9]: 'complic'

In [10]: PorterStemmer().stem('complicate')
Out[10]: 'complic'
```

Sane's STATS Academy of Statistics

Lemmatisation

- Lemmatisation 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.

```
In [46]: from nltk.stem.wordnet import WordNetLemmatizer

In [47]: from nltk.corpus import wordnet

In [48]: lemmatizer = WordNetLemmatizer()

In [49]: print(lemmatizer.lemmatize('going', wordnet.VERB))
go

In [50]: print(lemmatizer.lemmatize('went', wordnet.VERB))
go

In [51]: print(lemmatizer.lemmatize('gone', wordnet.VERB))
go

In [52]: print(lemmatizer.lemmatize('goes', wordnet.VERB))
go
```

Latent Semantic Analysis

- We should be creating a document x word matrix of the text we want to analyse
- If we have m documents then a document to word matrix can be created with documents mentioned in rows and words mentioned in columns. The cells in the table can be just occurrences / counts

Cat eats mice.
My cat is very ugly
Rat is ugly
Cat is very very shy

	cat	eats	mice	my	is	very	ugly	rat	shy
S1	1	1	1	0	0	0	0	0	0
S2	1	0	0	1	1	1	1	0	0
S3	0	0	0	0	1	0	1	1	0
S4	1	0	0	0	1	2	0	0	1

TF and IDF Weight

- The TF-IDF weight is a weight often used in information retrieval and text mining.
- TF-IDF weight which represents the importance of a term inside a document. It does this by comparing the frequency of usage inside an individual document as opposed to the entire data set (a collection of documents).
- It is calculated by TF*IDF.
- TF-IDF weight is product of TF and IDF quantities.

Term Frequency

- The importance increases proportionally to the number of times a word appears in the individual document itself--this is called Term Frequency.

- TF(t,D) = (Number of times term t appears in a document D)

Inverse Document Frequency

- Inverse document frequency is computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.

- IDF(t) = log_e(Total number of documents / Number of documents with term t in it).

Latent Semantic Analysis

- LSA models typically replace raw counts in the document-term matrix with a tf-idf score
- Tf-idf, or term frequency-inverse document frequency, assigns a weight for term *j* in document *i* as follows:

$$TF - IDF_{i,j} = TF_{i,j} * log \frac{N}{df_j}$$

How to compute?

- Consider a document containing 100 words where the word *Shoes* appears 6 times. The term frequency (i.e., tf) for shoes is then (6 / 100) = 0.06. Now, if we have 10 million documents and the word *Shoes* appears in 1000 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: 0.06 * 4 = 0.24.