# Application of Big Data in Social Science

# week 6

Heidi Hyeseung Choi
Fall 2022
HYSIS
heidichoi@hanyang.ac.kr

# Announcements

| | | |
|---|---|---|
| ~~9.1~~ | ~~1~~ | ~~Introduction~~ |
| ~~9.8~~ | ~~2~~ | ~~Web Scraping~~ |
| ~~9.15~~ | ~~3~~ | |
| ~~9.22~~ | ~~4~~ | ~~Natural Language Processing~~ |
| ~~9.29~~ | ~~5~~ | |
| **10.6** | **6** | **Text Analysis**<br>(recorded lecture on week 7) |
| 10.13 | 7 | |
| 10.20 | 8 | Mid term exam (as school schedule) |
| 10.27 | 9 | Social Network Analysis |
| 11.3 | 10 | |
| 11.10 | 11 | Machine Learning:<br>Supervised Learning |
| 11.17 | 12 | |
| 11.24 | 13 | Machine Learning:<br>Unsupervised Learning |
| 12.1 | 14 | |
| 12.8 | 15 | Data Visualization |
| 12.15 | 16 | Final Exam |

**No class on WEEK 7!**
**Recorded lecture will be available next week (Probably Sunday 9th)**

We will cover… Sentiment analysis and Word clouds!

Steps for making word clouds

1. download data
2. extract the text document
3. process our text data
   - tokenization
   - POS tagging
   - lemmatization
4. make and save word clouds!

# Announcements

- Mid term exam: October 20th 1:00pm – 2:15pm
- On mid term exam:

1. LMS portion: PDF lecture notes (50 points)
   - Multiple choice : 3 marks or 4 marks
   - True or false: 2 marks

2. Coding portion: Jupyter Notebook (50 points)
   - Coding: 2-4 marks

   = Total 100

Will be using both LMS online exam system and jupyter notebook.

My tentative plan

| | points | # | worth |
|---|---|---|---|
| LMS | 2 | 5 | 10 |
| | 3 | 8 | 24 |
| | 4 | 4 | 16 |
| CODING | 2 | 4 | 8 |
| | 3 | 6 | 18 |
| | 4 | 6 | 24 |
| Total | | | 100 |

# On mid term exam

- On the exam day, only laptops are allowed.
- I will provide you with
  - Blank A4 sheet
  - Documentation papers (ONLY FOR JUPYTER NOTEBOOK PORTION, not LMS)

NOT allowed:

- Internet search (googling, naver, python.org etc.,)
- Email, chat and instant messaging programs
- smart phones
- When placing the mobile phone in a visible place
- Cheating
- Flicking through PDF files (alt+tab) ^^

If you cheat or do something suspicious or do what is listed as not allowed:

- Score '0' will be processed for the mid term exam **without any exception.**

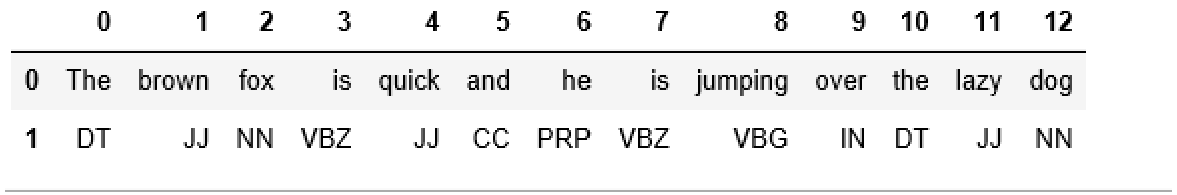
# Recap from last week

# Order of NLP

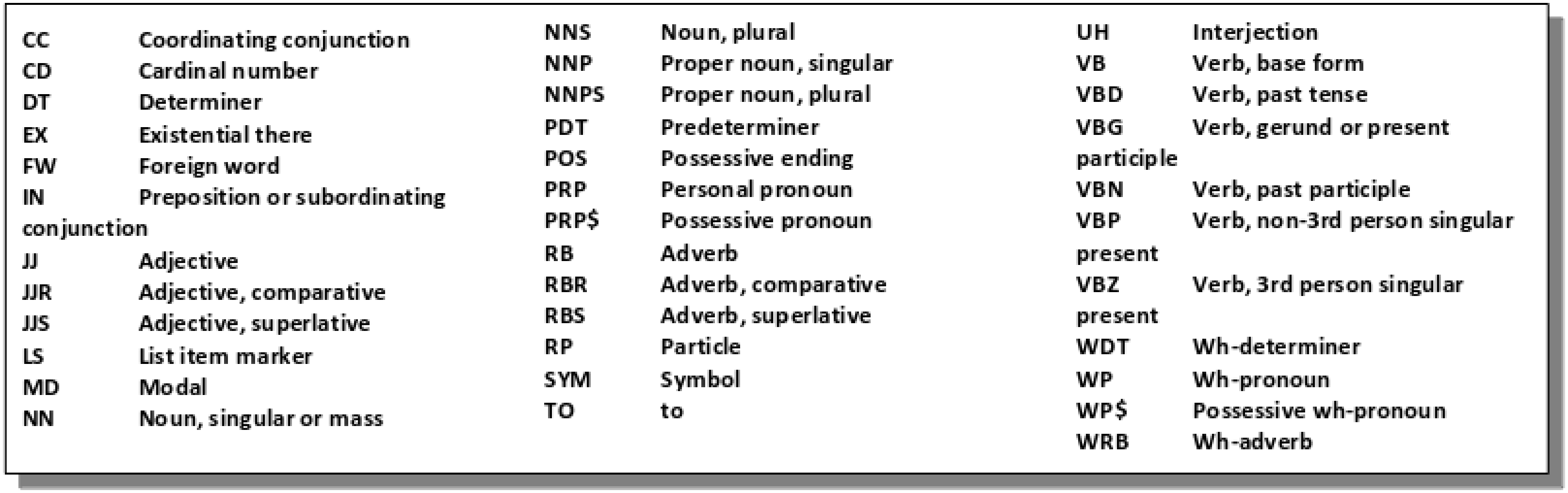
- HTML
- HTML parsing
- sentence segmentation
- word tokenization
- stopwords
- POS tagging
- text lemmatization

# Words :POS tagging

- Words are the smallest unit in a language.
- It is useful to annotate and tag words then analyze them into their POS to see the major syntactic categories.



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | The | brown | fox | is | quick | and | he | is | jumping | over | the | lazy | dog |
| 1 | DT | JJ | NN | VBZ | JJ | CC | PRP | VBZ | VBG | IN | DT | JJ | NN |

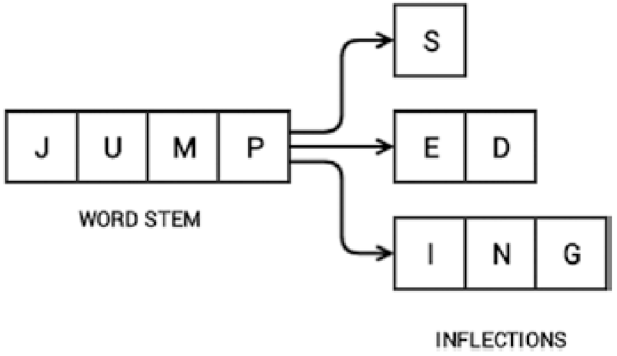


| | |
|---|---|
| CC | Coordinating conjunction |
| CD | Cardinal number |
| DT | Determiner |
| EX | Existential there |
| FW | Foreign word |
| IN | Preposition or subordinating conjunction |
| JJ | Adjective |
| JJR | Adjective, comparative |
| JJS | Adjective, superlative |
| LS | List item marker |
| 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 |
| PRP$ | Possessive pronoun |
| RB | Adverb |
| RBR | Adverb, comparative |
| RBS | 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, non-3rd person singular present |
| VBZ | Verb, 3rd person singular present |
| WDT | Wh-determiner |
| WP | Wh-pronoun |
| WP$ | Possessive wh-pronoun |
| WRB | Wh-adverb |

# Chunking

- The process of extracting phrases, or ‘chunks’ of texts from unstructured text.
- A common group of chunking is the noun phrase chunk (NP chunk).
- POS tags are used to create chunk grammar.
- Regex is used to make chunk grammar.

# Word stemming



- Word stem is the base form of a word where we can create new words by attaching affixes to them.
- There are different algorithms, such as Porter Stemmer, Snowball Stemmer, Lancaster Stemmer etc.
- **Porter Stemmer:** The Porter stemming algorithm

  (or "Porter stemmer") uses suffix-stemming to produce stems.
- **Snowball Stemmer:** Upgraded version, a.k.a. Porter2 Stemmer.
- **Lancaster Stemmer:** Compared to snowball and porter stemming, lancaster is the most aggressive stemming algorithm because it tends to over-stem a lot of words. It tries to reduce the word to the shortest stem possible.

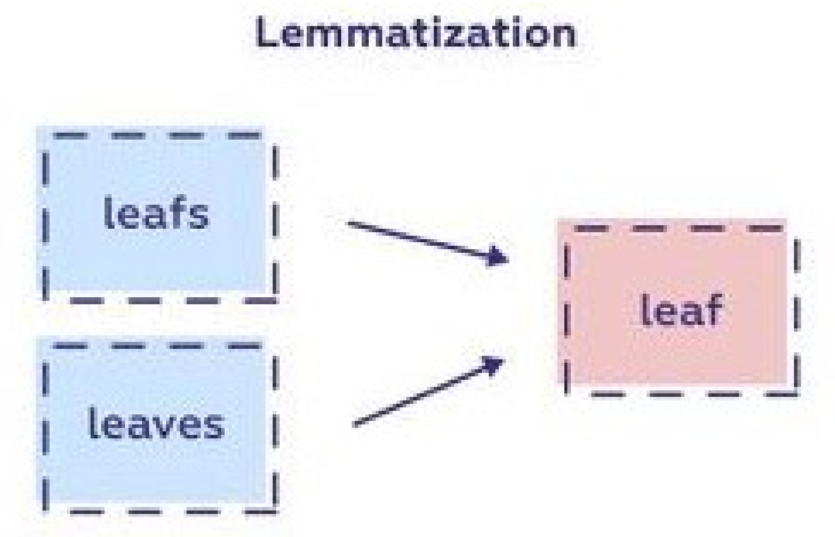
=> Stemming is fast, it is not 100% accurate.

**Note!**

The purpose of the Porter stemmer is not to produce complete words but to find variant forms of a word.

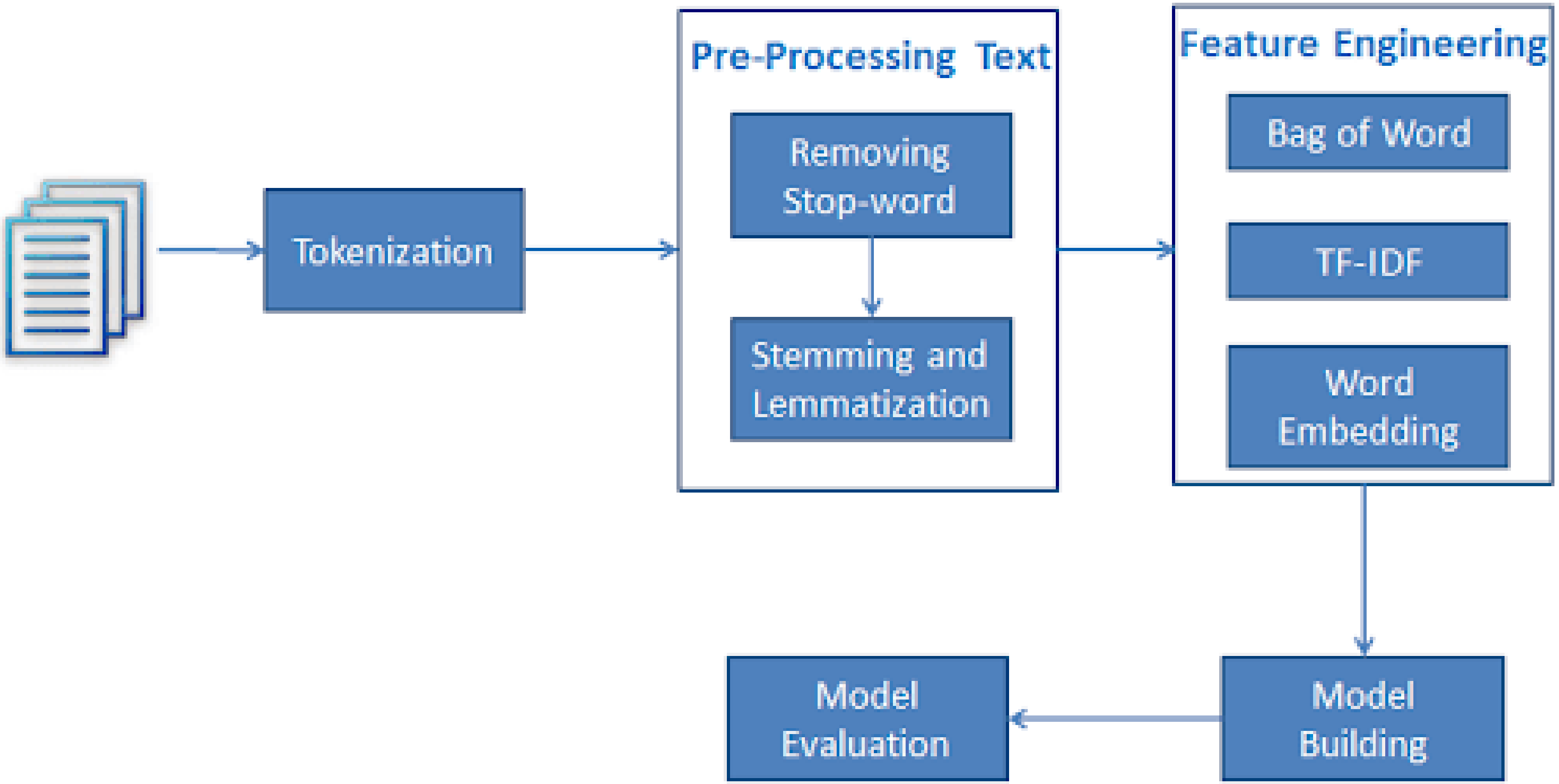
# Lemmatization

- Lemmatization is similar to stemming, but base form for lemmatization is the root word.
- In Lemmatization, the parts of speech(POS) will be determined first, unlike stemming which stems the word to its root form without considering the context.



- Lemmatization always considers the context and converts the word to its meaningful root/dictionary(WordNet) form called Lemma.
- It is important to do POS tagging before using this algorithm or it would assume every word as a noun.
- we will use NLTK package module 'WordNet' for lemmatization.

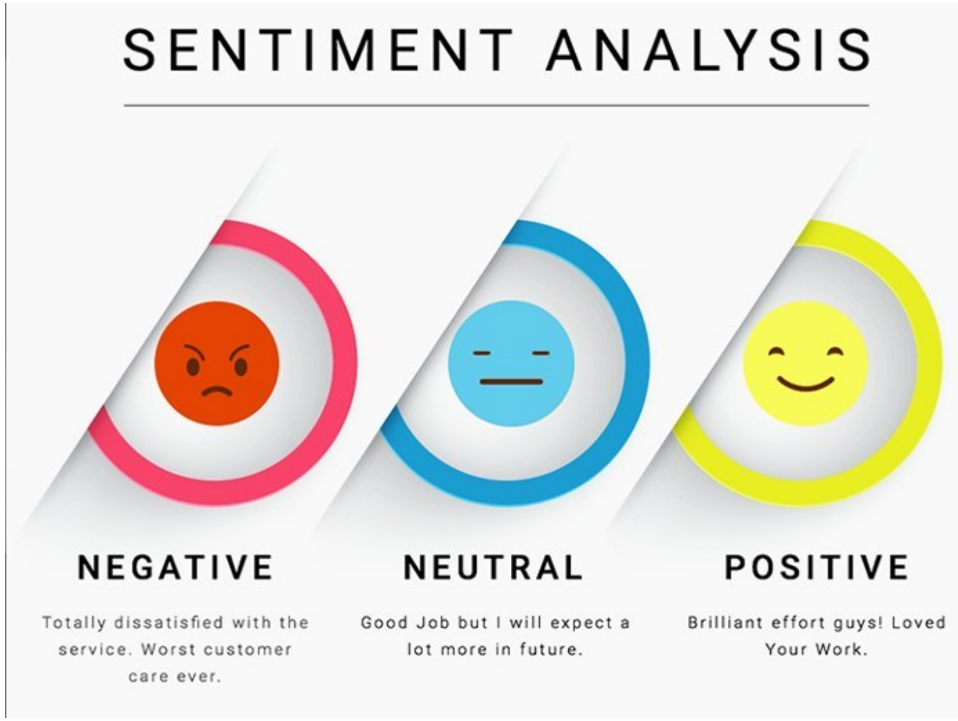
# What we went through so far this semester..



# Text analysis

- Text classification:
  - Bag of Word (BoW) model
  - TF-IDF model

- Sentiment analysis
  - Subjectivity
  - Polarity


SENTIMENT ANALYSIS
NEGATIVE
Totally dissatisfied with the service. Worst customer care ever.
NEUTRAL
Good Job but I will expect a lot more in future.
POSITIVE
Brilliant effort guys! Loved Your Work.

# Bag-of-Words model

- A bag of words is a representation of text that describes the occurrence of words within a document.
- We just keep track of word counts and disregard the grammatical details and the word order.
- It is called a "*bag*" of words, because any information about the order or structure of words in the document is discarded.
- The model is only concerned with whether known words occur in the document, not *where* in the document.
- BoW depends on absolute term frequencies.

⇒ It is one of the simplest feature extraction* technique used for text data.

This is a good day.
Is this a good day?

**Feature?**

- Column represents a measurable piece of data can be used for analysis.

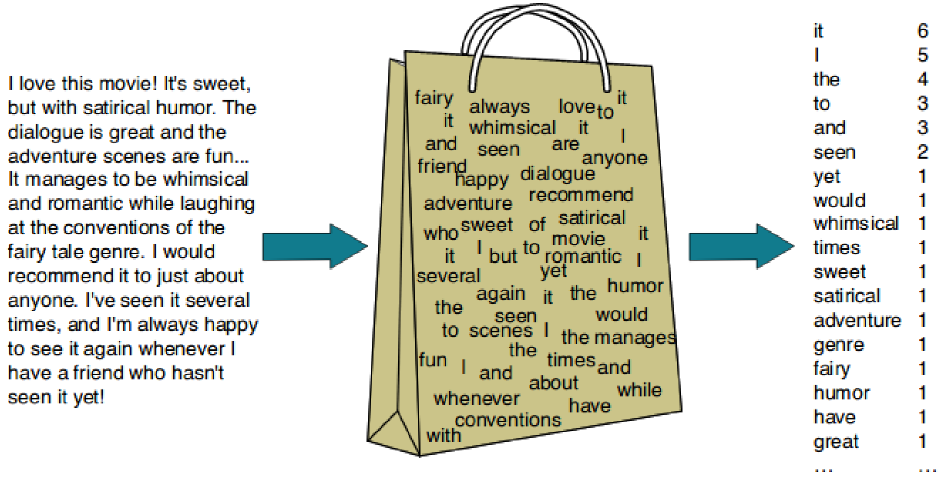
Eg) Name, age, income etc. (Variable)

**Feature extraction?**

The process of transforming raw data into numerical features that can be processed while preserving the information in the original data set.

# Bag-of-Words model

- **1) Tokenize** each document and give an integer id for each token.
- **2) Count** the occurrences of tokens in each document.
- **3) Convert** into numeric vector so that each document is represented by one vector in the feature matrix and each column represents a unique word.
- Result in a sparse matrix containing many 0s



I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

| | |
|---|---|
| it | 6 |
| I | 5 |
| the | 4 |
| to | 3 |
| and | 3 |
| seen | 2 |
| yet | 1 |
| would | 1 |
| whimsical | 1 |
| times | 1 |
| sweet | 1 |
| satirical | 1 |
| adventure | 1 |
| genre | 1 |
| fairy | 1 |
| humor | 1 |
| have | 1 |
| great | 1 |
| ... | ... |

sentence 1(or document 1) : The weather is nice today.

sentence 2(or document 2) : The weather seems to be quite cold today.

| | The | weather | is | nice | today | seems | to | be | quite | cold |
|---|---|---|---|---|---|---|---|---|---|---|
| Doc 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| Doc 2 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

# Bag-of-Words model Pros and Cons

Pros:

- Simple and easy to use.
- Can be used to create an initial draft model.

Cons:

- Does not consider semantic relations between words.
  - You can use word2vec n-grams etc models
- It results in sparse matrix (highly dependent on the number of unique words) => require a lot of memories
- Highly frequent words start to dominate in the document but may not be so informative.

So.. We can turn to a little more complicated technique (TF-IDF)

- TF-IDF is useful in solving the major drawbacks of Bag of words by introducing an important concept called **inverse document frequency(IDF).**
- It reflects how important a word is to a document in a collection of documents (a.k.a corpus)
- *how many times does this word appear in this document among the number of times all words appear in this document?*

# TF-IDF Model (Term Frequency-Inverse Document Frequency)

- TF: the number of occurrences of a specific term in a document.
  - It indicates how important a specific term in a document.
- IDF: how common or rare a word is in the entire document set.
  - It is the weight of a word, it aims to reduce the weight of a word if the word's occurrences are scattered throughout all the documents.
  - Works by taking the total number of documents, dividing it by the number of documents that contains a word, and apply logarithmic scaling to the result.
  - The closer to 0, the more common the word.

$$idf_i = \log(\frac{n}{df_i})$$

-Where $idf_i$ is the IDF score for word $i$,<br>
-$df_i$ is the number of documents containing word $i$,<br>
-$n$ is the total number of documents.<br>
-The higher the DF* of a word, the lower the IDF for the term.<br>
-When the number of DF is equal to n which means that the term appears in all documents, the IDF will be zero (log(1)=0)<br>
-We use log to weigh down the frequent terms while scaling up the rare ones.

*Document Frequency(DF) is the number of documents containing a specific word.

# TF-IDF Model (Term Frequency-Inverse Document Frequency)

$$w_{i,j} = tf_{i,j} \times idf_i$$

- Where $w_{ij}$ is TF-IDF score for word *i* in document *j*,
- $tf_{ij}$ is term frequency for word *i* in document *j*, and $idf_i$ is IDF score for word i.

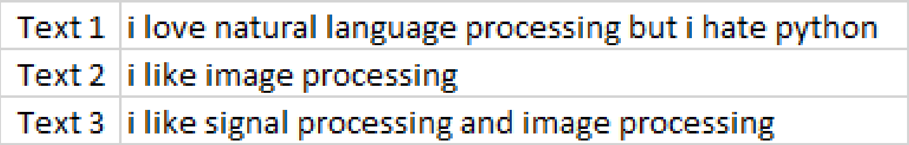
- TF-IDF: words that are common in every document, such as ‘the’, ’a’, ‘this’, ‘if’ ranks low(close to 0) even though they may appear many times.
- In other words, if word is very common and appears in many documents, the number would approach 0. Otherwise it would approach 1.
- Word with high tf-idf in a document, it is most of the times occurred in given documents and must be absent in the other documents, thus, ‘signature word’.

Usages?

- TF-IDF is useful extracting keywords from texts.
- The highest scoring words of a document are the most relevant to the document. Thus, considered as keywords.

# TF-IDF Model (Term Frequency-Inverse Document Frequency)

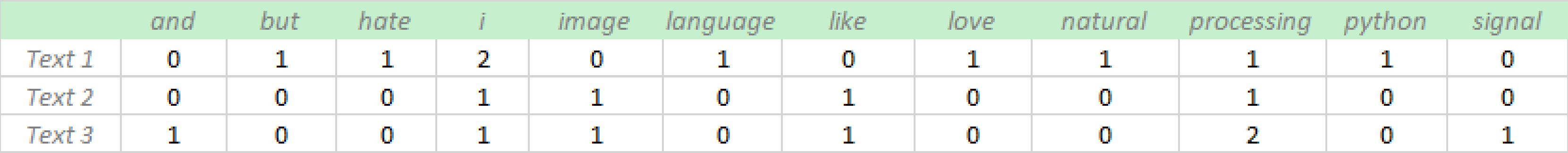
## Example



| | |
|---|---|
| Text 1 | i love natural language processing but i hate python |
| Text 2 | i like image processing |
| Text 3 | i like signal processing and image processing |

### Step 1

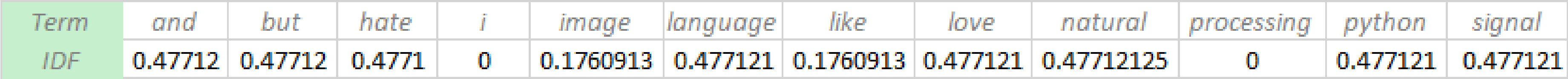
- Create a term frequency matrix where rows are documents and columns are distinct terms throughout all documents. Count word occurrences in every text.



| | and | but | hate | i | image | language | like | love | natural | processing | python | signal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Text 1 | 0 | 1 | 1 | 2 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| Text 2 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| Text 3 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 2 | 0 | 1 |

### Step 2

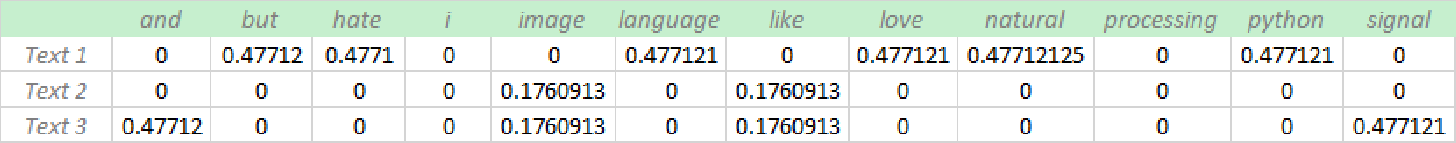
- Compute inverse document frequency (IDF) using the previously explained formula.
- The word *'I'* and '*processing'* has 0 IDF score.



| Term | and | but | hate | i | image | language | like | love | natural | processing | python | signal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IDF | 0.47712 | 0.47712 | 0.4771 | 0 | 0.1760913 | 0.477121 | 0.1760913 | 0.477121 | 0.47712125 | 0 | 0.477121 | 0.477121 |

### •Step 3

Multiply TF matrix with IDF respectively



| | and | but | hate | i | image | language | like | love | natural | processing | python | signal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Text 1 | 0 | 0.47712 | 0.4771 | 0 | 0 | 0.477121 | 0 | 0.477121 | 0.47712125 | 0 | 0.477121 | 0 |
| Text 2 | 0 | 0 | 0 | 0 | 0.1760913 | 0 | 0.1760913 | 0 | 0 | 0 | 0 | 0 |
| Text 3 | 0.47712 | 0 | 0 | 0 | 0.1760913 | 0 | 0.1760913 | 0 | 0 | 0 | 0 | 0.477121 |

https://towardsdatascience.com/tf-idf-simplified-aba19d5f5530

We will continue with sentiment analysis
in the next class
(recorded lecture)

# Let’s go to jupyter notebook!

```
>>> pip install textblob
>>> pip install sklearn
```

**If you get module not found error try below.**

```
>>> pip install -U textblob
>>> python –m textblob.download_corpora
```

# What is Sentiment analysis?

- Sentiment: a view of or attitude toward a situation or event; an opinion.
- Sentiment analysis: the use of NLP, text analysis to systematically identify, extract, quantify and study affective states and subjective information.
- Polarity: quantifying sentiment with positive or negative value.
- Subjectivity: generally refers to personal opinion, emotions or judgments.
- usages:
    - monitoring and measuring sentiment for social media posts and reviews
    - enhance customer experience
    - market research etc

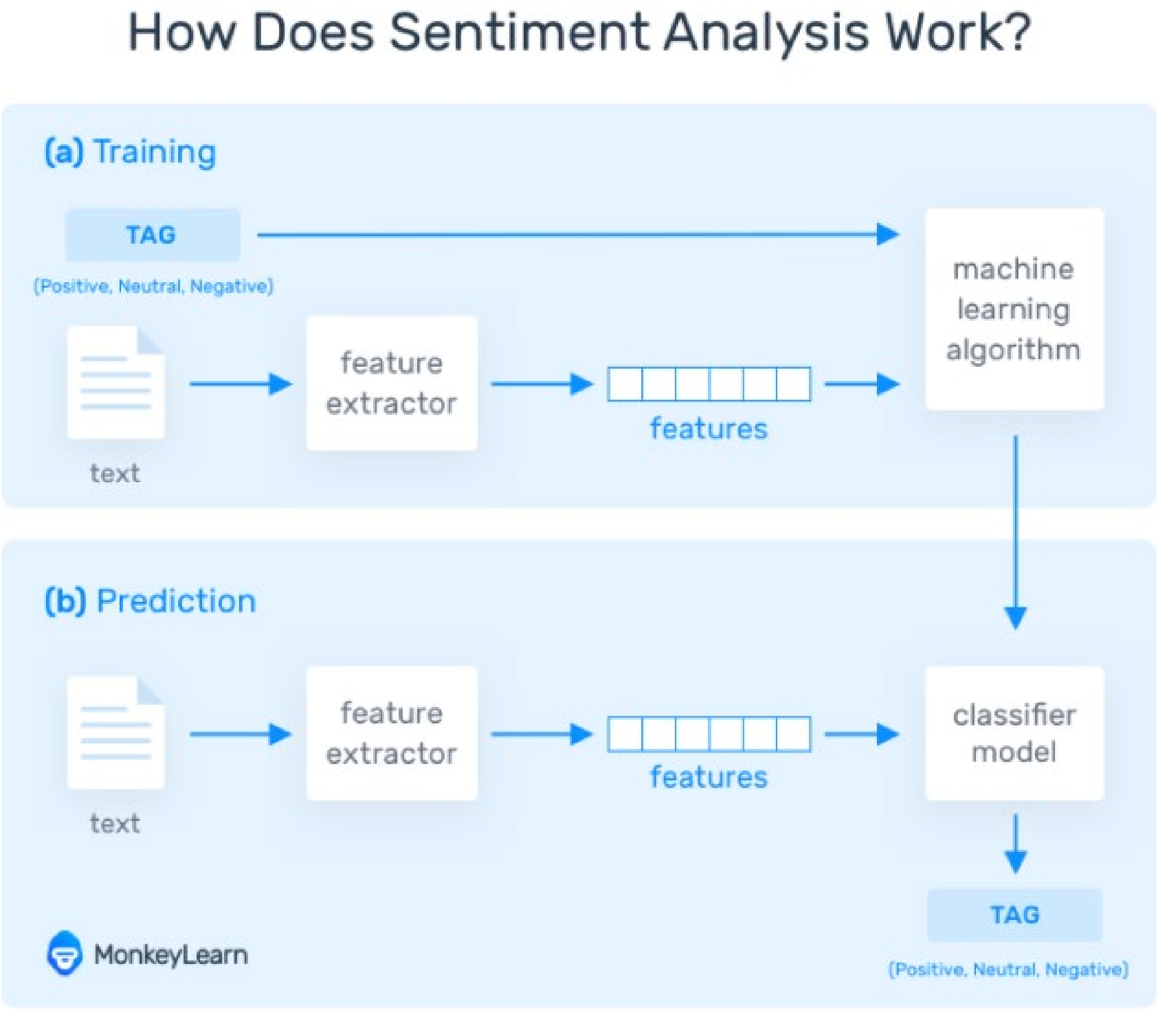
# Techniques for sentiment analysis

1. Lexicon based (Rule-based) sentiment analysis
   1. define lists of polarized words.
   2. count the number of positive and negative words in a given text.
   3. Compare the appearances of both, and calculate.

2. Machine Learning (Automatic) based sentiment analysis
   1. split the data set into a training set and a test set
   2. transform text into numerical feature vectors
   3. train the model
   4. evaluate the model
   5. predict new data.

# Fyi: Machine Learning (Automatic) based sentiment analysis

- feature:
  - a measurable property of the object that you are trying to analyze.
  - in datasets features appear as columns.


How Does Sentiment Analysis Work?
(a) Training
TAG
(Positive, Neutral, Negative)
text
feature extractor
features
machine learning algorithm
(b) Prediction
text
feature extractor
features
classifier model
TAG
(Positive, Neutral, Negative)
MonkeyLearn

# TextBlob library

- Sentiment analysis is basically the process of determining the attitude or the emotion of the writer, i.e., whether it is positive or negative or neutral.
- The *sentiment* function of textblob returns two properties, **polarity**, and **subjectivity**.
- Polarity returns a float which lies in the range of [-1,1] where 1 means positive statement and -1 means a negative statement.
- Subjectivity quantifies the amount of personal opinion and factual information contained in the text. The higher subjectivity means that the text contains personal opinion rather than factual information.
- Subjectivity is also a float which lies in the range of [0,1].

# Python basics for today’s work

# Join() Method for String

- The join() method takes all items in an iterable and joins them into one string.

syntax

```
>>> string.join(iterable)
```

Examples:

```
>>> names = [‘heidi’, ‘john’, ‘jenny’]
>>> x = “-”.join(names)
>>> print(x)
>>> heidi-john-jenny
>>> y= “ ”.join(names)
>>> print(y)
```

# isalnum(), isalpha(), isdigit() methods in strings

- isalnum()
  - method returns ‘TRUE’ if all the characters are alphanumeric (alphabet (a-z) and numbers(0-9)
  - Not alphanumeric: (space)!(#$*etc
- isalpha()
  - method returns ‘TRUE’ if all the characters are alphabet letters (a-z).
- isdigit()
  - method returns ‘TRUE’ if all the characters are digits, otherwise False.
- syntax

```
>>> string.isalnum()
>>> x = ‘hello123’
>>> x.isalnum()
>>> TRUE
```

```
>>> x.isalpha()
>>> FALSE
>>> x.isdigit()
>>> FALSE
```

# lambda function

- Similar to for loops.
- “*A lambda function is a small function containing **a single expression**. Lambda functions can also act as anonymous functions where they don’t require any name. These are very helpful when we have to perform small tasks with less code.*”

- keyword

- bound variable/argument

- expression

```
lambda x: x
```

```
lambda x: x + 5


def add_5(x):
        return x+5
```

# lambda function + apply function in pandas

- apply() function in pandas calls the lambda function and applies to every row or column of the dataframe and returns a modified copy of the dataframe.

```
>>> df[‘height’]= df[‘height’].apply(lambda x: x+10)
>>> df[‘category’] = df[‘height’].apply(lambda x: ‘Adult’ if x>160 else ‘Child’)
```

| name | height |
|---|---|
| A | 125 |
| B | 175 |
| C | 185 |

| name | height | category |
|---|---|---|
| A | 135 | Child |
| B | 185 | Adult |
| C | 195 | Adult |