Text Classification

Table of Content

What will We Learn Today?

1. Handling Text Data (Tokenization, stopwords, stemming, lemmatization)
2. Feature Extraction (BOW and TF-IDF)
3. ML model for text classification

Dataset types

age	anaemia	creatinine_p	diabetes	ejection_f	high_bloo	platelets	serum_cre	serum_so	sex	smoking	time	DEATH_EVENT
75	0	582	0	20	1	265000	1.9	130	1	0	4	1
55	0	7861	0	38	0	263358	1.1	136	1	0	6	1
65	0	146	0	20	0	162000	1.3	129	1	1	7	1
50	1	111	0	20	0	210000	1.9	137	1	0	7	1
65	1	160	1	20	0	327000	2.7	116	0	0	8	1
90	1	47	0	40	1	204000	2.1	132	1	1	8	1
75	1	246	0	15	0	127000	1.2	137	1	0	10	1
60	1	315	1	60	0	454000	1.1	131	1	1	10	1

Heart failure clinical records

review	sentiment
One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked. T	positive
A wonderful little production. The filming technique is very unassuming- very old-time	positive
I thought this was a wonderful way to spend time on a too hot summer weekend, sitting in the air c	positive
Basically there's a family where a little boy (Jake) thinks there's a zombie in his closet & his parents	negative
Petter Mattei's "Love in the Time of Money" is a visually stunning film to watch. Mr. Mattei offers u	positive
Probably my all-time favorite movie, a story of selflessness, sacrifice and dedication to a noble cau	positive
I sure would like to see a resurrection of a up dated Seahunt series with the tech they have today it	positive
This show was an amazing, fresh & innovative idea in the 70's when it first aired. The first 7 or 8 yea	negative
Encouraged by the positive comments about this film on here I was looking forward to watching thi	negative
If you like original gut wrenching laughter you will like this movie. If you are young or old then you	positive
Phil the Alien is one of those quirky films where the humour is based around the oddness of everyt	negative
I saw this movie when I was about 12 when it came out. I recall the scariest scene was the big bird e	negative
So im not a big fan of Boll's work but then again not many are. I enjoyed his movie Postal (maybe im	negative
The cast played Shakespeare. Shakespeare lost. I appreciate that this is trying	negative

IMDB Movie Review

Cat and Dog dataset

Example

- IMDB dataset having 50K movie reviews for natural language processing or Text analytics.
- This is a dataset for binary sentiment classification
- Source : https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews?select=IMDB+Dataset.csv

review	sentiment
One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked. T	positive
A wonderful little production. The filming technique is very unassuming- very old-time-	positive
I thought this was a wonderful way to spend time on a too hot summer weekend, sitting in the air c	positive
Basically there's a family where a little boy (Jake) thinks there's a zombie in his closet & his parents	negative
Petter Mattei's "Love in the Time of Money" is a visually stunning film to watch. Mr. Mattei offers u	positive
Probably my all-time favorite movie, a story of selflessness, sacrifice and dedication to a noble cau:	positive
I sure would like to see a resurrection of a up dated Seahunt series with the tech they have today it	positive
This show was an amazing, fresh & innovative idea in the 70's when it first aired. The first 7 or 8 yea	negative
Encouraged by the positive comments about this film on here I was looking forward to watching thi	negative
If you like original gut wrenching laughter you will like this movie. If you are young or old then you	positive
Phil the Alien is one of those quirky films where the humour is based around the oddness of everyt	negative
I saw this movie when I was about 12 when it came out. I recall the scariest scene was the big bird e	negative
So im not a big fan of Boll's work but then again not many are. I enjoyed his movie Postal (maybe im	negative
The cast played Shakespeare. Shakespeare lost. I appreciate that this is trying	negative

Reading CSV dataset

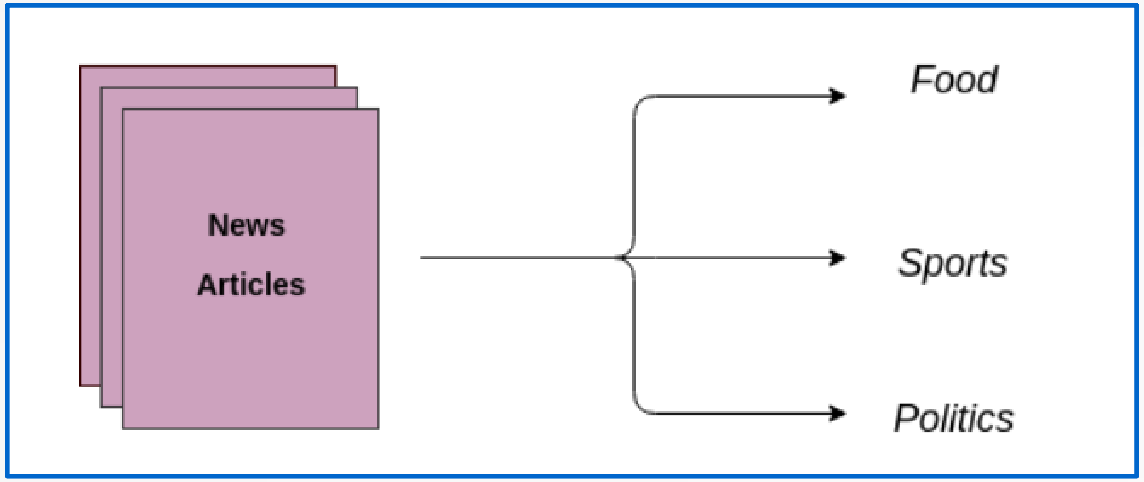
- Read the CSV file

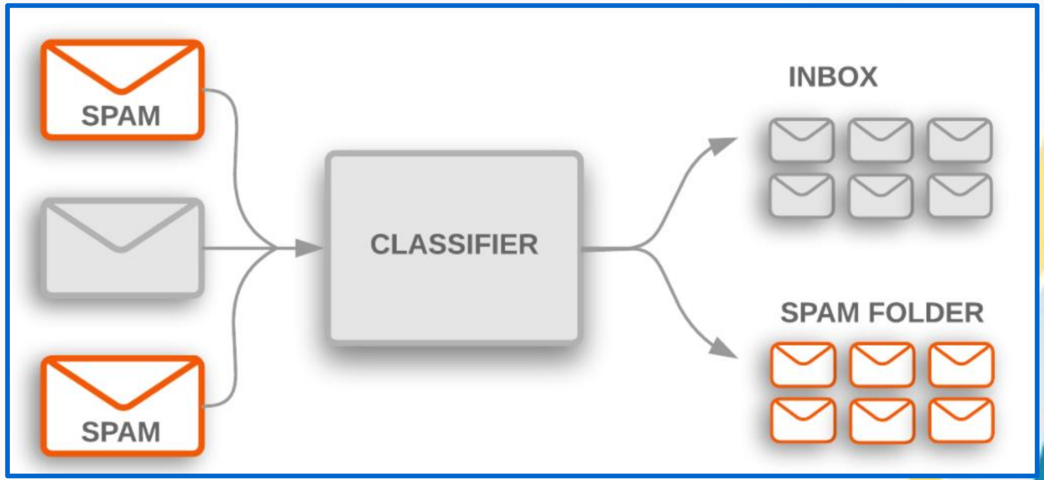
```
import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/ganjar87/data_science_practice/main/IMDB_small_size.csv', delimiter=',')
df.head()
```

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

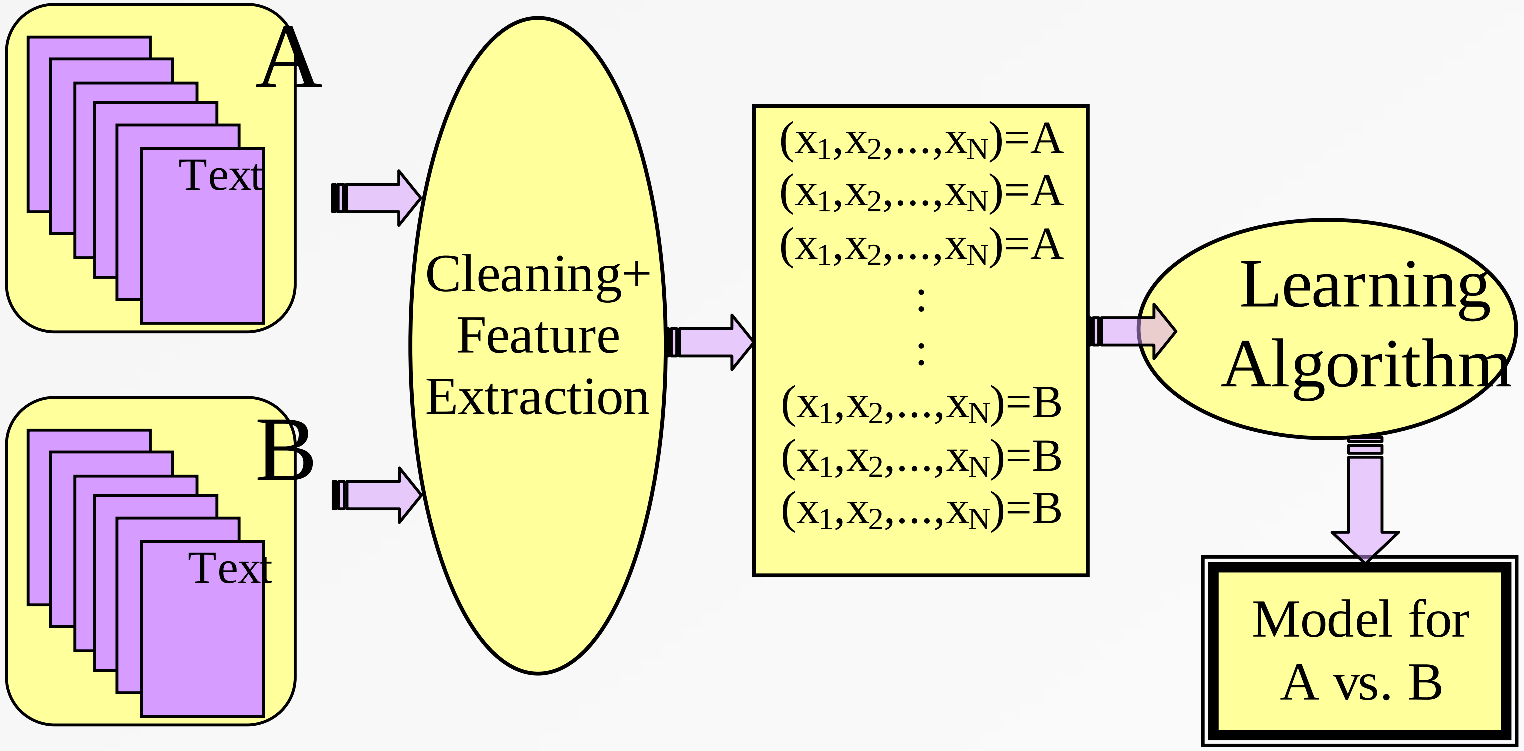
Text Classification

- *Text classification* juga dikenal sebagai *text tagging* atau *text categorization* adalah proses mengkategorikan teks ke dalam kelompok tertentu.
- *Text classification* adalah salah satu tugas dasar dalam *natural language processing (NLP)* dengan aplikasi yang luas contohnya *sentiment analysis, topic labeling, spam detection, dan intent detection*.



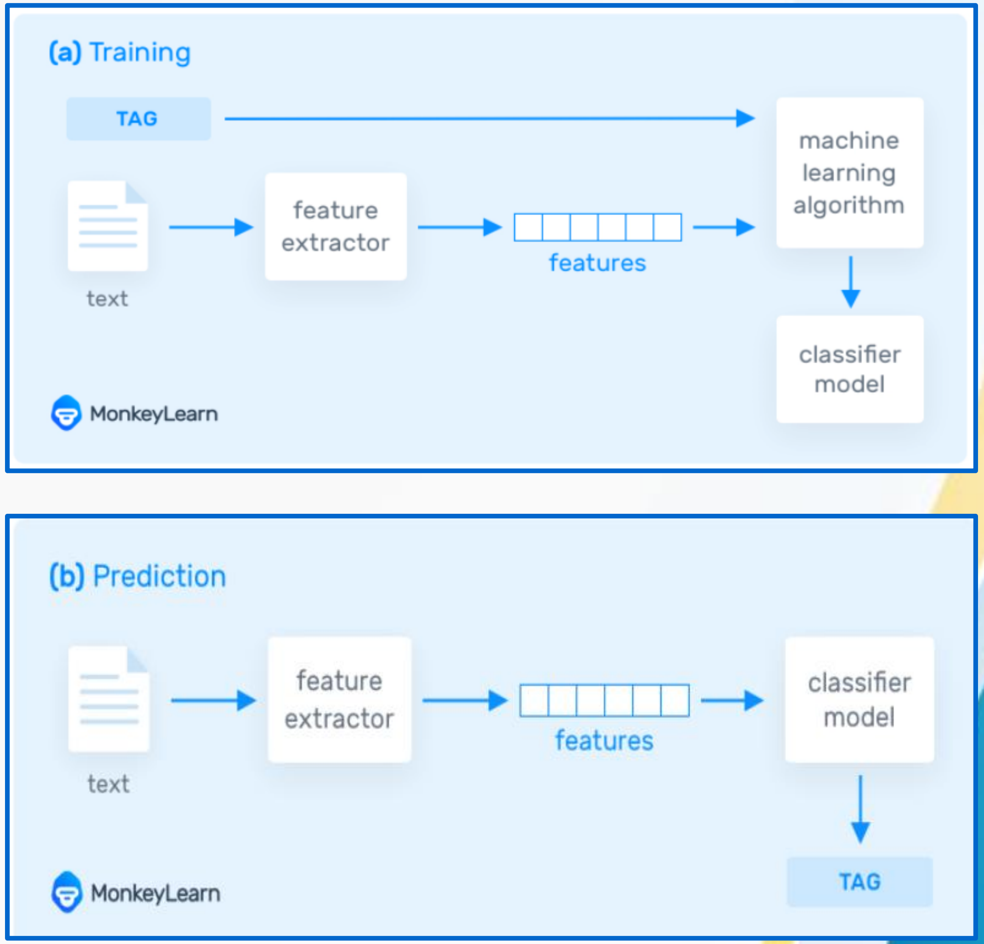


Text Classification



Text Classification

- Ada tiga pendekatan dalam *text classification*
- ***Rule-based System***
  - Teks dipisahkan ke dalam kelompok terorganisir menggunakan *handicraft linguistic rules.*
- ***Machine Learning-based System***
  - *ML-based classifier* membuat klasifikasi berdasarkan pengamatan sebelumnya dari kumpulan data
- ***Hybrid System***
  - Menggabungkan *machine learning classifier* dengan *rule-based system*, digunakan untuk meningkatkan performa.

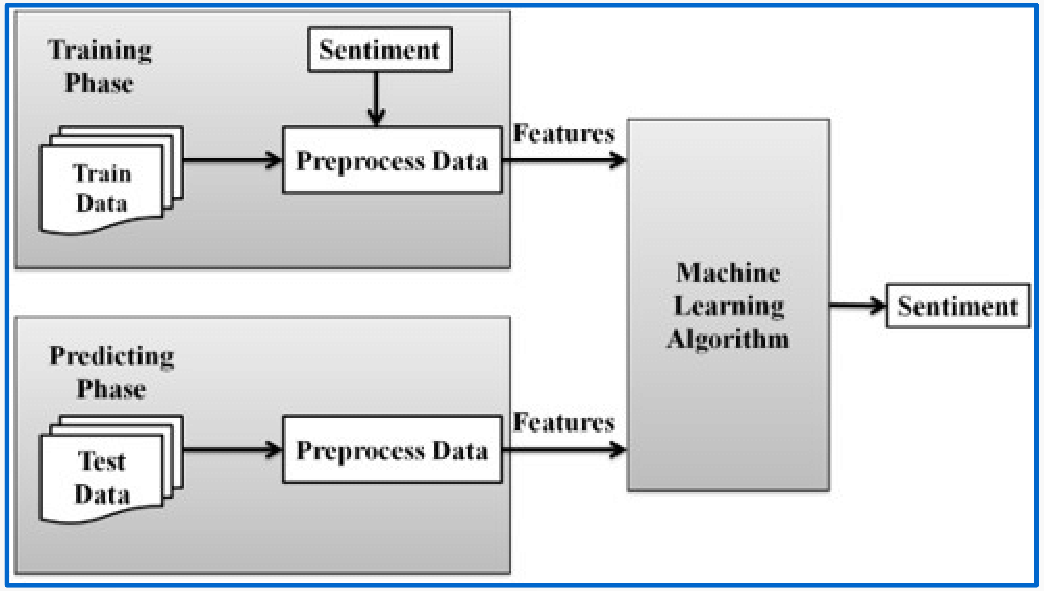


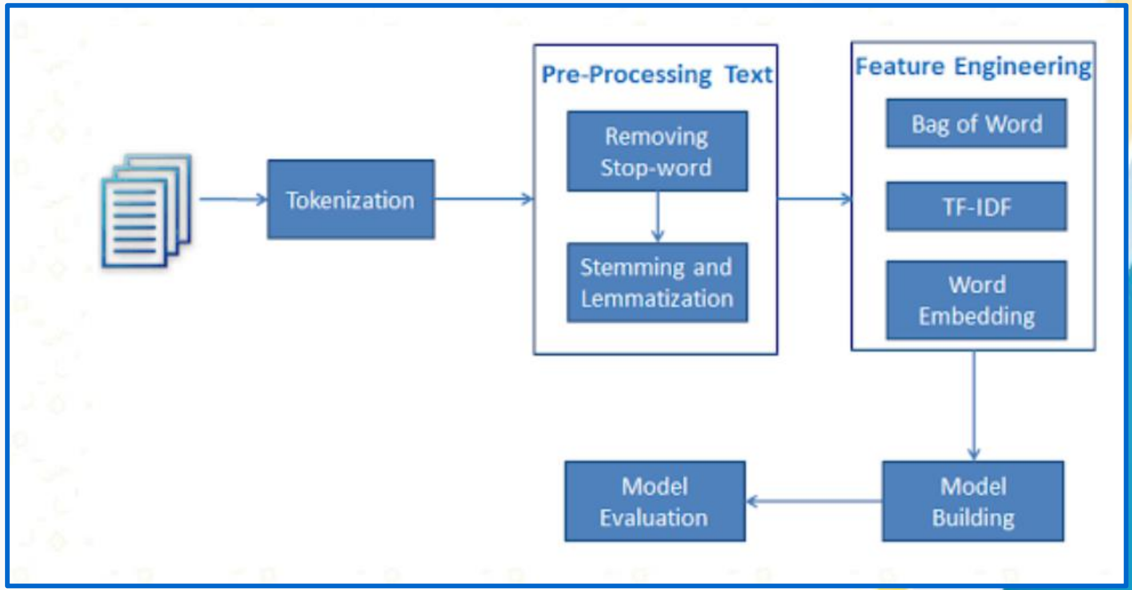
Machine Learning-based System

https://monkeylearn.com/text-classification/

Sentiment Analysis-Definition

- Salah satu contoh aplikasi dari *text classification* adalah *sentiment analysis.*
- Adalah metode yang secara otomatis memahami persepsi pelanggan terhadap suatu produk atau layanan berdasarkan komentar mereka.

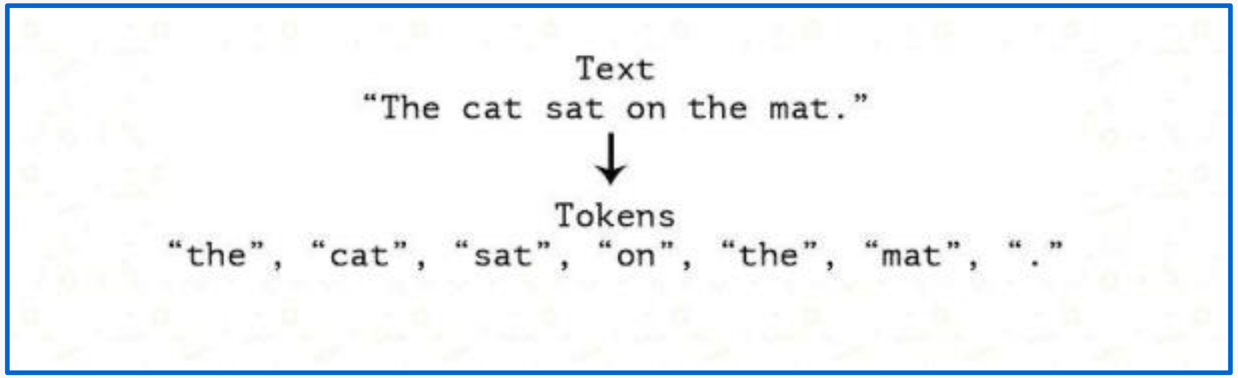




Handling text dataset

Tokenization

- Tokenization is breaking the raw text into small chunks.
- Tokenization breaks the raw text into words, sentences called tokens.
- These tokens help in understanding the context or developing the model for the NLP.



Tokenization

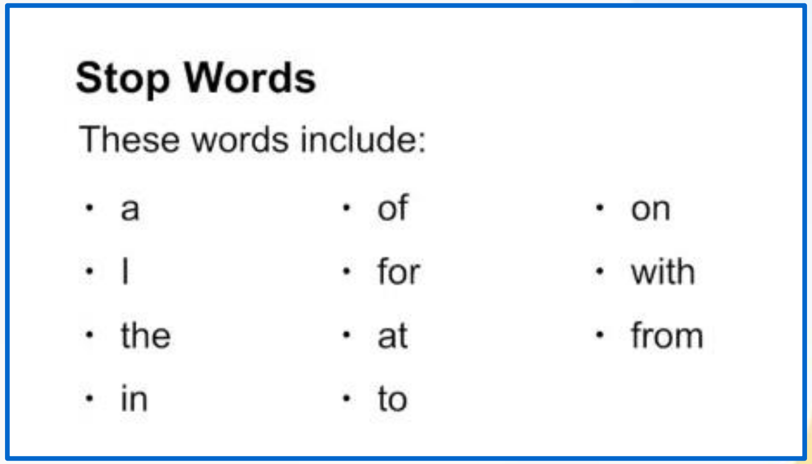
- Make sure that nltk library is downloaded
- We use word_tokenize function from nltk library

```
[23] import nltk
     nltk.download('punkt')
     from nltk.tokenize import word_tokenize

     df_resize = df[:1000]
     df_text = df_resize['review'].astype(str)
     df_class = df_resize['sentiment']
     lines = df_text.values.tolist()

     list_tokens = list()
     for line in lines:
       line = line.replace("<br />","")
       tokens = word_tokenize(line)
       list_tokens.append(tokens)

     [nltk_data] Downloading package punkt to /root/nltk_data...
     [nltk_data]   Package punkt is already up-to-date!
```

```
list_tokens[0]

['One',
 'of',
 'the',
 'other',
 'reviewers',
 'has',
 'mentioned',
 'that',
 'after',
 'watching',
 'just',
 '1',
 'Oz',
 'episode',
 'you',
 "'ll",
 'be',
 'hooked',
 '.',
 'They',
 'are',
 'right',
```

Pre-processing the Text

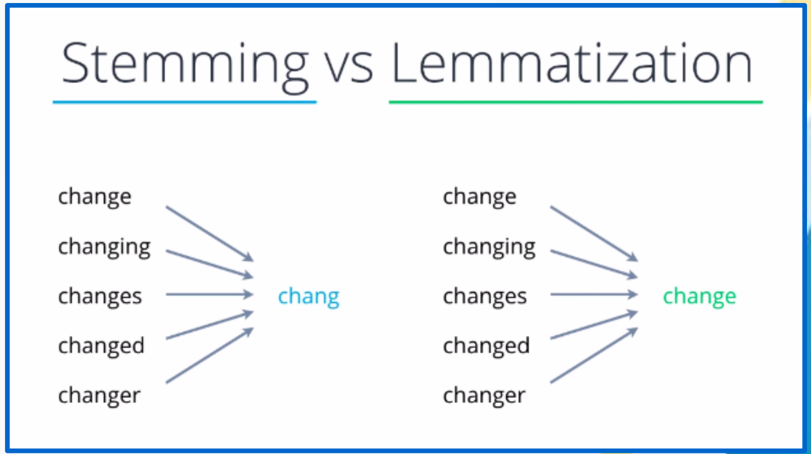
- Removing stop words
  - Punctuations
    - Example : .?””,’;:-[]()
  - Prepositions
    - Example : "in," "at," "on," "of," and "to."
- Stemming
  - Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form—generally a written word form.
    - Example : walker, walked, walking => walk
- Lemmatization
  - Lemmatization is the process of converting a word to its base form.
  - Converts the word to its meaningful base form



Stop Words

These words include:

- a
- I
- the
- in
- of
- for
- at
- to
- on
- with
- from



Remove stop words + stemming

- We use small size dataset (10 records only), to reduce computation time.

```python
import pandas as pd
nltk.download('stopwords')
#this one for lemmatization
#nltk.download('wordnet')
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer

df_resize = df[:10]
df_text = df_resize['review'].astype(str)
df_class = df_resize['sentiment']
lines = df_text.values.tolist()
list_tokens = list()
porter=PorterStemmer()
lemmatizer = WordNetLemmatizer()
for line in lines:
    line = line.replace("<br />","")
    tokens = word_tokenize(line)
    tokens = [w.lower() for w in tokens]
    #check alphabhet
    tokens = [word for word in tokens if word.isalpha()]
    #remove stopwords
    tokens = [word for word in tokens if not word in stopwords.words()]
    #stemming
    tokens = [porter.stem(word) for word in tokens]
    #append into list
    list_tokens.append(tokens)
```

```python
list_tokens[0]
```

```
 'oz',
 'mess',
 'around',
 'first',
 'episod',
 'ever',
 'saw',
 'struck',
 'nasti',
 'surreal',
 'could',
 'say',
 'readi',
 'watch',
 'develop',
 'tast',
 'oz',
 'got',
 'accustom',
 'high',
 'level',
 'graphic',
 'violenc',
 'violenc',
 'injustic',
 'crook',
 'guard',
 'sold',
 'nickel',
```

Remove stop words + lemmatization

- We use small size dataset (10 records only), to reduce computation time.

```
import pandas as pd
nltk.download('stopwords')
#this one for lemmatization
nltk.download('wordnet')
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer

df_resize = df[:10]
df_text = df_resize['review'].astype(str)
df_class = df_resize['sentiment']
lines = df_text.values.tolist()
list_tokens = list()
porter=PorterStemmer()
lemmatizer = WordNetLemmatizer()
for line in lines:
    line = line.replace("<br />","")
    tokens = word_tokenize(line)
    tokens = [w.lower() for w in tokens]
    #check alphabhet
    tokens = [word for word in tokens if word.isalpha()]
    #remove stopwords
    tokens = [word for word in tokens if not word in stopwords.words()]
    #stemming
    #tokens = [porter.stem(word) for word in tokens]
    #lemmatization
    tokens = [lemmatizer.lemmatize(word) for word in tokens]
    #append into list
    list_tokens.append(tokens)
```

Feature Extraction

Bag of Words

- Frequency of the term in the document
- We are only concerned with encoding schemes that represent what words are present, without any information about order

```
doc1 = "saya belajar pemrograman dan belajar melukis"
doc2 = "saya membantu adik saya belajar menulis"
doc3 = "ibu belajar menjahit"
```

adik	belajar	dan	ibu	melukis	membantu	menjahit	menulis	pemrograman	saya
0	2	1	0	1	0	0	0	1	1
1	1	0	0	0	1	0	1	0	2
0	1	0	1	0	0	1	0	0	0

Bag of Words

- First, combine the tokens into sentence
- Use CountVectorizer from sklearn

```
[32] from sklearn.feature_extraction.text import CountVectorizer


     new_doc = list()
     for doc in list_tokens:
       row= ' '.join(doc)
       new_doc.append(row)

     vectorizer = CountVectorizer(max_features=1000)
     X_input = vectorizer.fit_transform(new_doc)
```

```
print(X_input.toarray())
print(vectorizer.vocabulary_)
print(vectorizer.get_feature_names())
```

```
[[1 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 1 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 1 0]]
{'reviewer': 371, 'mentioned': 278, 'watching': 496, 'oz': 310, 'episode': 133, 'hooked': 206, 'right': 373, 'exactly': 137,
['accustomed', 'acting', 'action', 'actor', 'addiction', 'adrian', 'agenda', 'agreement', 'air', 'aired', 'almost', 'amazing
```

TF-IDF

- Term frequency–inverse document frequency,
- Numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus
- TF-IDF are word frequency scores that try to highlight words that are more interesting, e.g. frequent in a document but not across documents.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

For a term i in document j:

tf_{ij} = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

idf(t) = log [n / df(t)] + 1

sklearn formula

```
doc1 = "saya belajar pemrograman dan belajar melukis"
doc2 = "saya membantu adik saya belajar menulis"
doc3 = "ibu belajar menjahit"
```

adik	belajar	dan	ibu	melukis	membantu	menjahit	menulis	pemrograman	saya
0	2	2.0986123	0	2.0986	0	0	0	2.09861229	1.405465
2.098612	1	0	0	0	2.09861229	0	2.0986123	0	2.81093
0	1	0	2.0986	0	0	2.09861229	0	0	0

TF-IDF

- First, combine the tokens into sentence
- User TfidfVectorizer from sklearn

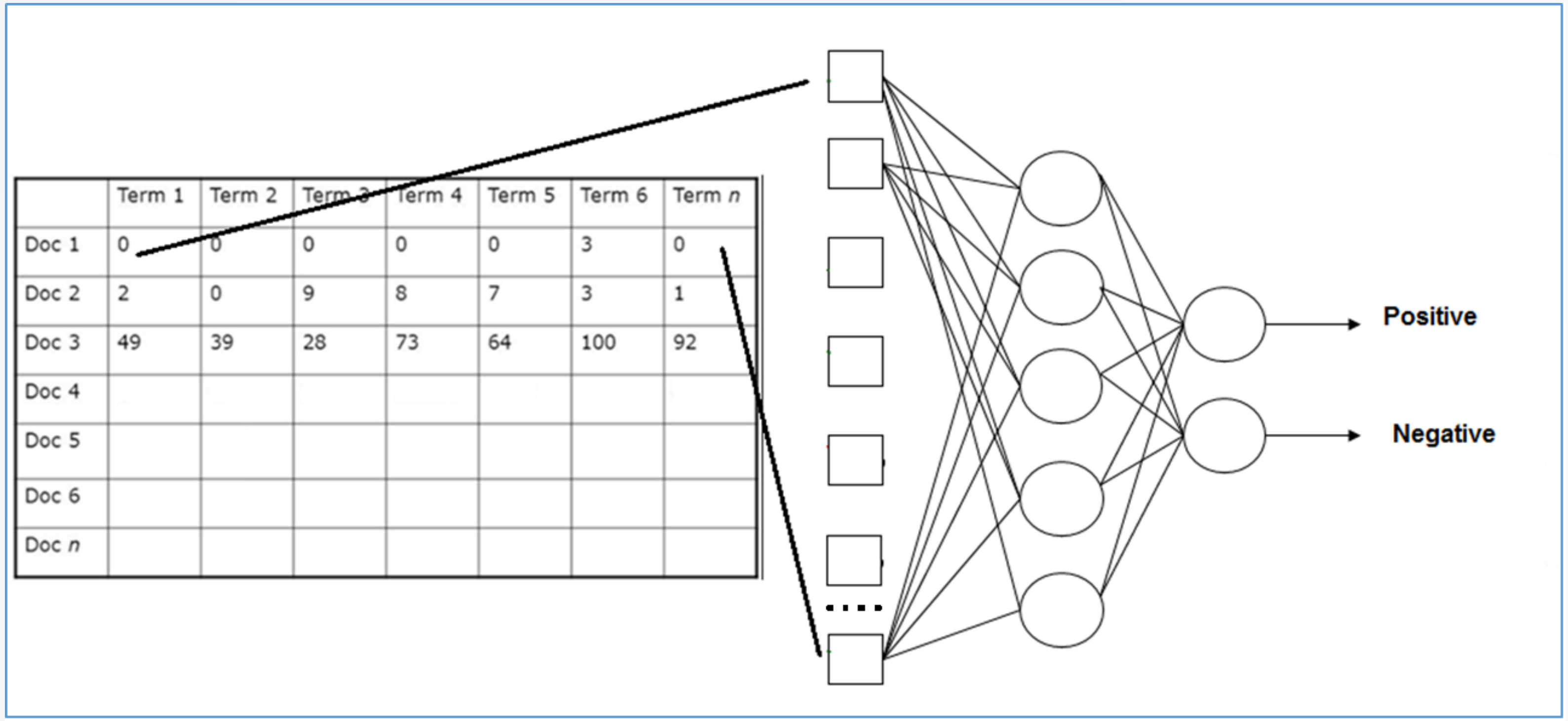
```
[39] from sklearn.feature_extraction.text import TfidfVectorizer
     new_doc = list()
     for doc in list_tokens:
       row= ' '.join(doc)
       new_doc.append(row)

     tfidf_vectorizer = TfidfVectorizer(max_features=1000)
     X_input = tfidf_vectorizer.fit_transform(new_doc)
```

```
print(tfidf_vectorizer.get_feature_names())
print(X_input.toarray())
print(tfidf_vectorizer.vocabulary_)
```

```
['accustomed', 'acting', 'action', 'actor', 'addiction', 'adrian', 'agenda', 'agreement', 'air', 'aired', 'al
[[0.06681545 0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.09669583 0.        ]
 ...
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.2263006  0.        ]]
{'reviewer': 371, 'mentioned': 278, 'watching': 496, 'oz': 310, 'episode': 133, 'hooked': 206, 'right': 373,
```

Sentiment analysis using MLP

- Use X_train as features for MLP

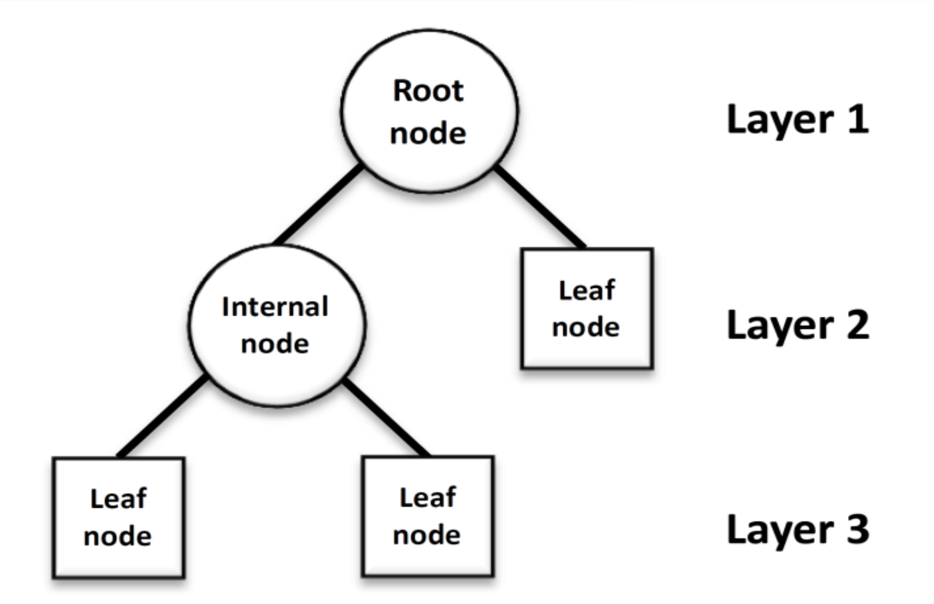


Discussion on NN

- Keuntungan
  - **Robust** -berfungsi baik ketika training set mengandung error
  - Output bisa discrete, real-valued, atau vector
- Kekurangan
  - Waktu yang lama saat training
  - Sulit untuk dipahami

Decision Tree Induction

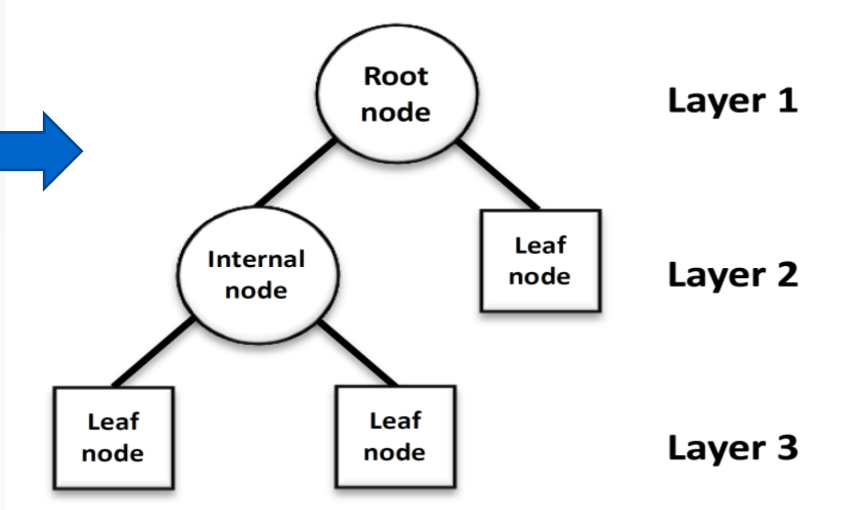
- Basic algorithm
  1. At start, all the training examples are at the root
  2. Test **attributes are selected** on the basis of a heuristic or statistical measure (e.g., information gain)
  3. **Examples are partitioned** recursively based on selected attributes



Sentiment analysis using DT

- Use X_train as features for DT

	Term 1	Term 2	Term 3	Term 4	Term 5	Term 6	Term *n*	class
Doc 1	0	0	0	0	0	3	0	
Doc 2	2	0	9	8	7	3	1	
Doc 3	49	39	28	73	64	100	92	
Doc 4								
Doc 5								
Doc 6								
Doc *n*								



Discussion on DT

- Kelebihan
  - Dapat diubah menjadi aturan klasifikasi yang dapat dipahami
  - Relatif cepat
- Kekurangan
  - Sensitive (not robust) terhadap noises
  - Continuous-valued attributes - partisi secara dinamis nilai atribut kontinu ke dalam set interval diskrit

Thank YOU