Pandas and
Data handling: text data

ECE30007 Intro to AI Project

Contents

- Introduction
- Pandas
- NLP: Natural Language Processing
- Exercises

pandas

- **Pandas?**
  - an open-source Python package that is most widely used for data science/data analysis and machine learning tasks
- **Advantage?**
  - Less writing and more work done
  - Excellent data representation
  - Made for Python
  - An extensive set of features

```
For installation

> pip install pandas
```

For API reference please visit *https://pandas.pydata.org/pandas-docs/stable/reference/index.html*

pandas – Data structure

- Series
  - one-dimensional labeled array capable of holding any type
  - **axis labels** are referred to as **index**

```
In [3]: import pandas as pd
        pd.__version__
```

```
Out[3]: '1.1.0'
```

```
In [4]: data = [1, 2, 3, 4]
        index = ['a', 'b', 'c', 'd']
        pd.Series(data=data, index=index)
```

```
Out[4]: a    1
        b    2
        c    3
        d    4
        dtype: int64
```

pandas – Data structure

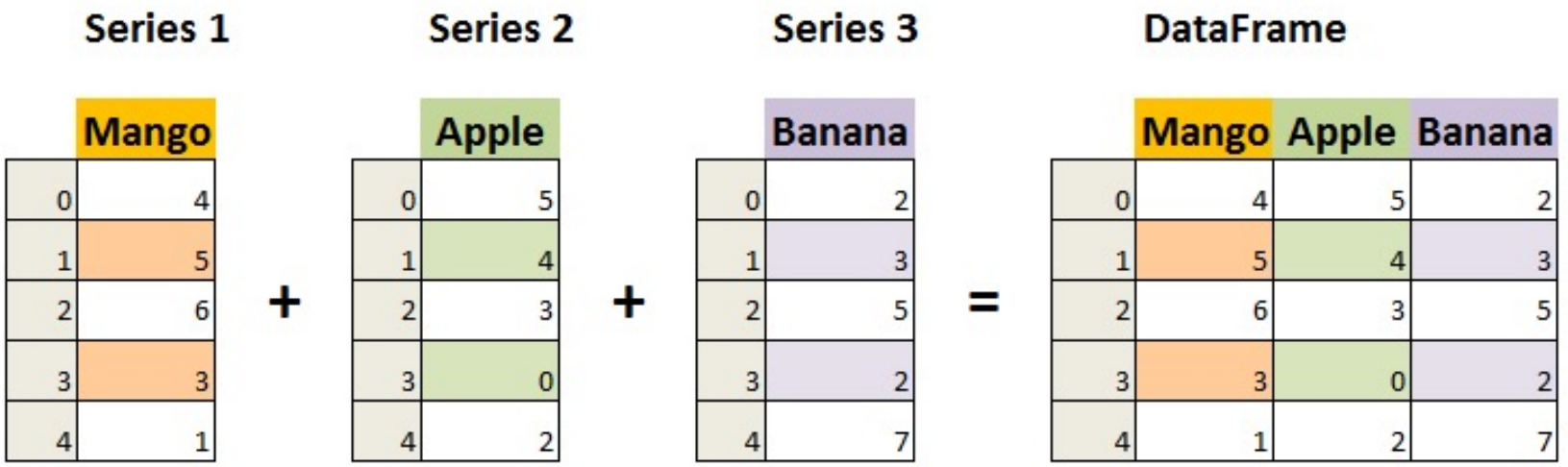
- DataFrame
  - 2-dimensional labeled data structure with columns of different types

In [6]:
```
data = {'col1':[1, 2, 3, 4], 'col2':['a', 'b', 'c', 'd']}
index = ['A', 'B', 'C', 'D']
pd.DataFrame(data=data, index=index)
```

Out[6]:

	col1	col2
A	1	a
B	2	b
C	3	c
D	4	d

Two common data structures in Pandas



https://www.datasciencemadesimple.com/create-series-in-python-pandas/

Exercise data

- Titanic Data
  - Provide information on the fate of passengers on the Titanic, summarized according to economic status(class), sex, age and survival

P. Id	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.25		S
2	1	1	Cumings, Mrs. John Bradley (Florence Brigg	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2.	7.925		S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1	C123	S
5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.05		S
6	0	3	Moran, Mr. James	male		0	0	330877	8.4583		Q
7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463	51.8625	E46	S
8	0	3	Palsson, Master. Gosta Leonard	male	2	3	1	349909	21.075		S
9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmin	female	27	0	2	347742	11.1333		S
10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14	1	0	237736	30.0708		C

Exercise data

- **Data information**
  - survived : Survival(0=dead, 1=survived)
  - pclass : Passenger Class(1 = 1st, 2 = 2nd, 3 = 3rd)
  - sibsp : Number of Siblings/Spouses Aboard
  - Parch : Number of Parents/Children Aboard
  - ticket : Ticket number
  - Fare : passenger Fare(British pound)
  - Cabin : Cabin (room number)
  - embarked : Port of Embarkation

    (C = Cherbourg; Q = Queenstown; S = Southampton)

pandas – Create and Load DataFrame

- Create DataFrame

```
In [7]: import numpy as np
        data = np.ones((5,3))
        pd.DataFrame(data=data)
```

Out[7]:

	0	1	2
0	1.0	1.0	1.0
1	1.0	1.0	1.0
2	1.0	1.0	1.0
3	1.0	1.0	1.0
4	1.0	1.0	1.0

- Load ‘data in csv file’ into DataFrame

```
In [11]: df = pd.read_csv('titanic_train.csv', index_col='PassengerId')
         df.head(1)
```

Out[11]:

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S

index_col = 'PassengerId' will use ‘PassengerId’ column in csv as column index in DataFrame.

pandas – Checking data information

- head & tail
  - Returns the first n rows

```
In [12]: df.head(2)
```

Out[12]:

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

```
In [22]: df.tail(2)
```

Out[22]:

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C
891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	Q

- sample
  - Returns a random sample of items from an axis of object

```
In [23]: df.sample(2)
```

Out[23]:

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
307	1	1	Fleming, Miss. Margaret	female	NaN	0	0	17421	110.8833	NaN	C
875	1	2	Abelson, Mrs. Samuel (Hannah Wizosky)	female	28.0	1	0	P/PP 3381	24.0000	NaN	C

pandas
- Descriptive statistics and aggregation methods

- describe

```
In [13]: df.describe()
```

Out[13]:

	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

- corr

```
In [14]: df.corr()
```

Out[14]:

	Survived	Pclass	Age	SibSp	Parch	Fare
Survived	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

pandas
- Descriptive statistics and aggregation methods

- Statistic method
  - mean

```
In [15]: df.mean()

Out[15]: Survived     0.383838
         Pclass       2.308642
         Age         29.699118
         SibSp        0.523008
         Parch        0.381594
         Fare        32.204208
         dtype: float64
```

  - groupby

```
In [16]: df.groupby("Pclass").mean()
```

Out[16]:

Pclass	Survived	Age	SibSp	Parch	Fare
1	0.629630	38.233441	0.416667	0.356481	84.154687
2	0.472826	29.877630	0.402174	0.380435	20.662183
3	0.242363	25.140620	0.615071	0.393075	13.675550

```
In [17]: df.groupby(['Survived', 'Pclass']).mean()
```

Out[17]:

Survived	Pclass	Age	SibSp	Parch	Fare
0	1	43.695312	0.287500	0.300000	64.684008
	2	33.544444	0.319588	0.144330	19.412328
	3	26.555556	0.672043	0.384409	13.669364
1	1	35.368197	0.492647	0.389706	95.608029
	2	25.901566	0.494253	0.643678	22.055700
	3	20.646118	0.436975	0.420168	13.694887

pandas - Selecting data

- loc
  - loc is label based indexing

```
In [14]: df.loc[0:5, "Name"]
```

```
Out[14]: PassengerId
1                                Braund, Mr. Owen Harris
2    Cumings, Mrs. John Bradley (Florence Briggs Th...
3                                 Heikkinen, Miss. Laina
4         Futrelle, Mrs. Jacques Heath (Lily May Peel)
5                               Allen, Mr. William Henry
Name: Name, dtype: object
```

- iloc
  - iloc is integer-location(from 0 to length-1) based indexing

```
In [16]: df.iloc[-5:, :5]
```

Out[16]:

PassengerId	Survived	Pclass	Name	Sex	Age
887	0	2	Montvila, Rev. Juozas	male	27.0
888	1	1	Graham, Miss. Margaret Edith	female	19.0
889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN
890	1	1	Behr, Mr. Karl Howell	male	26.0
891	0	3	Dooley, Mr. Patrick	male	32.0

pandas - Selecting data

- Dictionary approach

```
In [20]: df[['Age', 'Survived']]
```

Out[20]:

PassengerId	Age	Survived
1	22.0	0
2	38.0	1
3	26.0	1
4	35.0	1
5	35.0	0
...	...	...
887	27.0	0
888	19.0	1

- Boolean indexing

```
In [19]: mask = (df.Survived == 0)
         df[mask][:5]
```

Out[19]:

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S

pandas - Dataframe manipulation

- Add column

```
In [25]: # if you assign scalar, broadcasting
         df['new'] = 10
         df.head(2)
```

Out[25]:

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	new
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	10
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	10

```
In [26]: df['family'] = df['SibSp'] + df['Parch']
         df.head()
```

Out[26]:

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	new	family
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	10	1
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	10	1

pandas - Dataframe manipulation

- Drop columns

```
In [13]: df.head(1)
```

Out[13]:

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	new	family
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S	10	1

```
In [14]: ## using inplace
         # 'inplace' means there is no return value
         df.drop(columns=['new', 'family'], inplace=True)

In [15]: df.head(1)
```

Out[15]:

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S

Inplace = True will make changes within df.

pandas - Dataframe manipulation

- Concatenate DataFrame
  - Attach DataFrame or Series together

```
1 data = np.zeros(len(df))
2 index = np.array(range(1,len(df)+1))
3
4 new_ser= pd.Series(data, name = 'zeros', index=index)
5 new_df = pd.concat(objs=[df, new_ser], axis=1)
6
7 new_df.head(5)
8
```

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	family	zeros
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	1	0.0
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	1	0.0
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	0	0.0
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	1	0.0
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	0	0.0

pandas - Dataframe manipulation

- Rename
  - Alter axes labels.
  - ***inplace :*** whether to return a new DataFrame or apply the changes to itself

```
In [24]: df.columns

Out[24]: Index(['Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch', 'Ticket',
                'Fare', 'Cabin', 'Embarked'],
               dtype='object')

In [26]: df.rename(columns = {"Age":"How old"}, inplace=True)
         df.columns

Out[26]: Index(['Survived', 'Pclass', 'Name', 'Sex', 'How old', 'SibSp', 'Parch',
                'Ticket', 'Fare', 'Cabin', 'Embarked'],
               dtype='object')
```

pandas - Dataframe manipulation

- reset_index*(drop = False, inplace = False)*
  - Reset the index of the DataFrame, and use the default one instead.
  - ***drop:*** Do not try to insert index into dataframe columns. This resets the index to the default integer index.

```
In [43]: df
```

Out[43]:

PassengerId	Survived	Pclass	Name	Sex	How old	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...

```
In [44]: df_temp = df.copy()
         df_temp.reset_index(drop = True) # if False, it doesn't drop PassengerId
```

Out[44]:

	Survived	Pclass	Name	Sex	How old	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...

pandas - Dataframe manipulation

- fillna & isnull
  - **fillna :** Fill NA/NaN values using the specified method.
  - **isnull :** Detect missing values.

```
In [47]: df[df['How old'].isnull()]
```

Out[47]:

PassengerId	Survived	Pclass	Name	Sex	How old	SibSp	Parch	Ticket	Fare	Cabin	Embarked
6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
18	1	2	Williams, Mr. Charles Eugene	male	NaN	0	0	244373	13.0000	NaN	S
20	1	3	Masselmani, Mrs. Fatima	female	NaN	0	0	2649	7.2250	NaN	C
27	0	3	Emir, Mr. Farred Chehab	male	NaN	0	0	2631	7.2250	NaN	C

```
In [53]: print('number of null data in Age column: ', len(df[df['How old'].isnull()]))

         age_mean = df['How old'].mean()
         df['How old'].fillna(value=age_mean, inplace=True) # fill null value with age_mean

         print('number of null data in Age column: ', len(df[df['How old'].isnull()]))
```

```
number of null data in Age column:  177
number of null data in Age column:  0
```

pandas - element-wise function

- apply
  - Apply a function along an axis of the DataFrame.
  - What is *lambda?*
    - a small anonymous function with no name
    - it can take any number of arguments, but can only have one expression.

```
x = lambda a: a + 10
print(x(5))
```

```
15
```

pandas - element-wise function

- apply

	Survived	Pclass	Name	Sex	How old	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S

```
df['Sex']=df['Sex'].apply(lambda x: 0 if x == 'male' else 1)
df.head(3)
```

PassengerId	Survived	Pclass	Name	Sex	How old	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	0	22.0	1	0	A/5 21171	7.2500	NaN	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38.0	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	1	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S

- to_csv

```
In [76]: df.to_csv('titanic_tocsv.csv')
```

Exercise (1) number of survivals

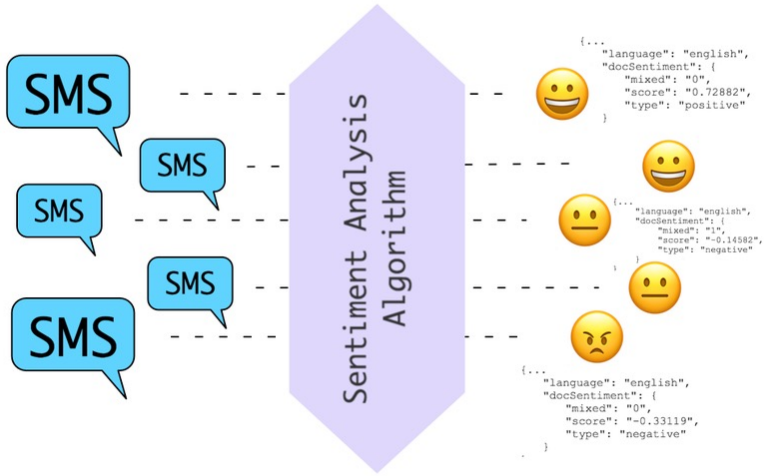
- Count the number of survivals per sex and passenger class.

```
group: ('female', 1) => 91
group: ('female', 2) => 70
group: ('female', 3) => 72
group: ('male', 1) => 45
group: ('male', 2) => 17
group: ('male', 3) => 47
```

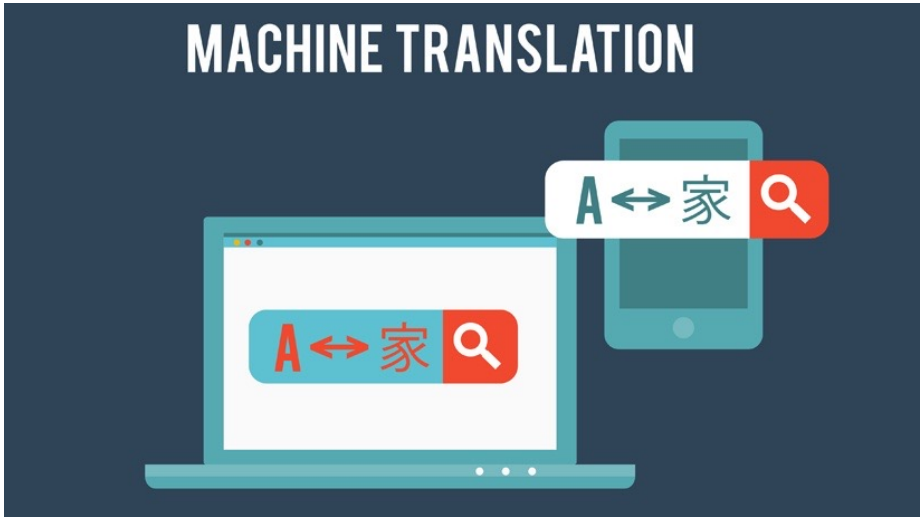
- Count the number of dead (non-survivals) per sex and passenger class.

```
group: ('female', 1) => 3
group: ('female', 2) => 6
group: ('female', 3) => 72
group: ('male', 1) => 77
group: ('male', 2) => 91
group: ('male', 3) => 300
```

What is NLP?

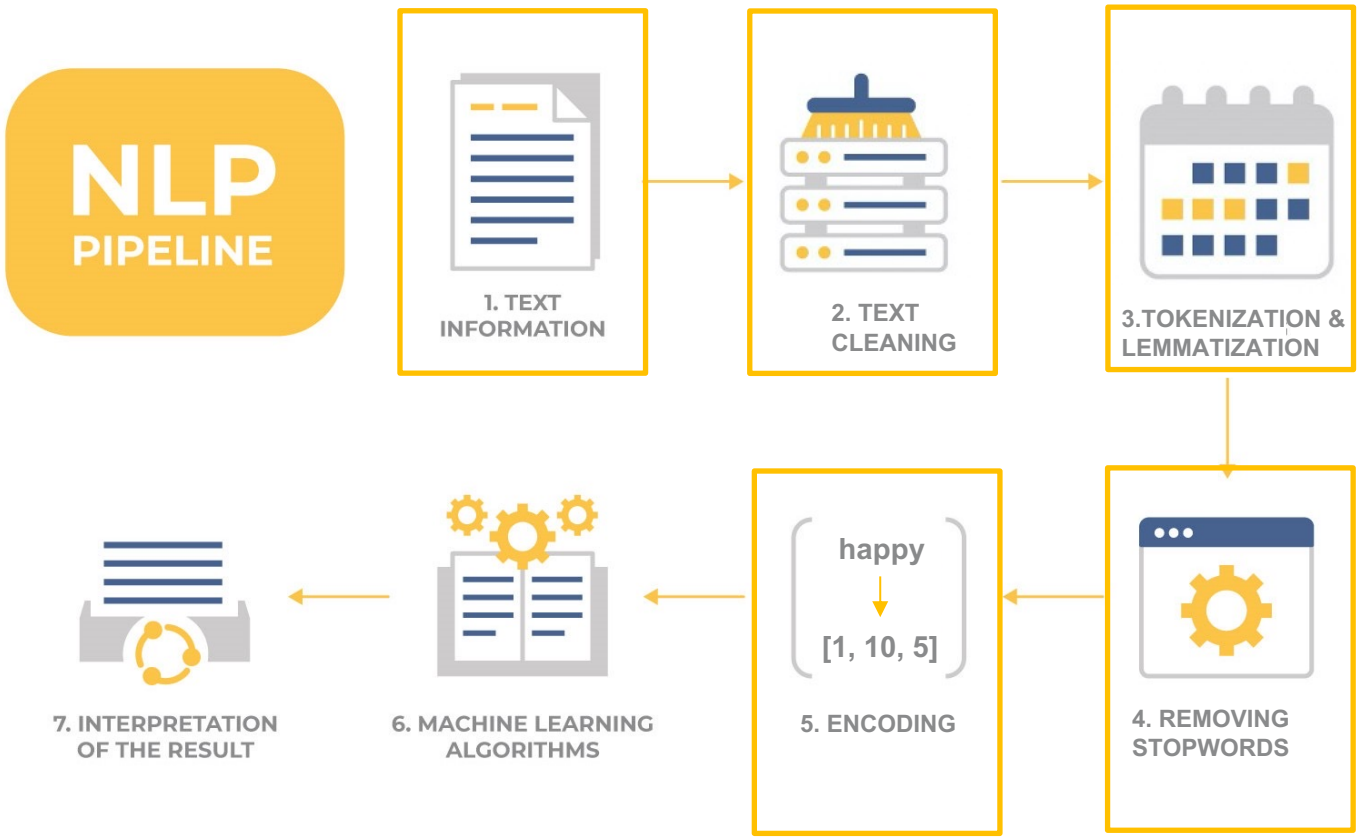


https://www.kaggle.com/aadilsrivastava01/a-beginners-guide-to-sentiment-analysis



https://elearningindustry.com/4-machine-translation-tools-incorporating-machine-translation

NLP pipeline



I'm studying COMPUTER SCINECE...!!

I'm studying computer science

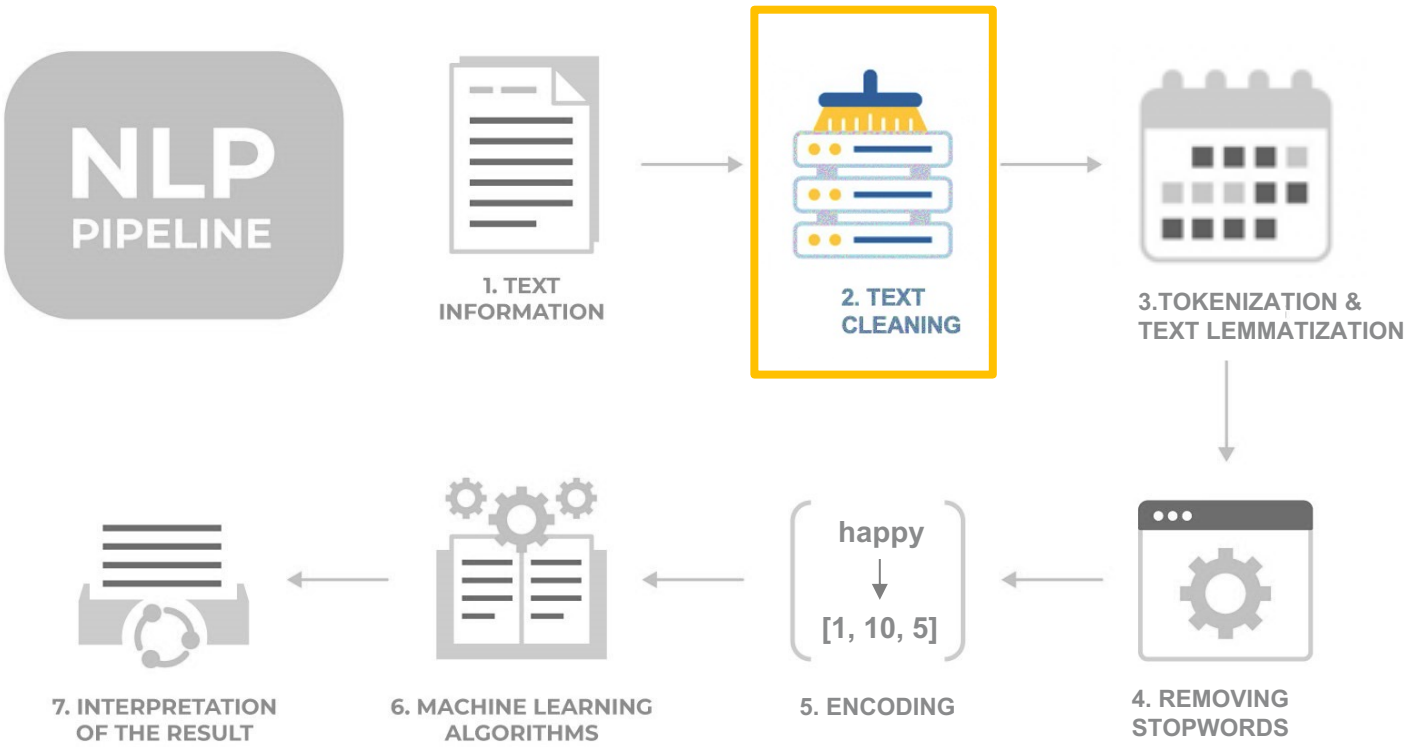
I | 'm | studying | computer | science

I | be | study | computer | science

study | computer | science

study	10	1.2, 0.1
computer	20	0.3, 2.1
science	30	-0.2, 1.2

NLP preprocessing



We need packages

```
> pip install nltk
> pip install BeautifulSoup4
> pip install lxml
> pip install spacy
> python -m spacy download en_core_web_sm
```

1. Text data

'<html><h2>What is nlp??? </h2></html> \nNatural Language Processing, or NLP for short, is broadly defined as the automatic manipulation of natural language, like speech and text, by software.\nThe study of natural language processing has been around for more than 50 years and grew out of the field of linguistics with the rise of computers.\n(In this post), you will discover what natural language processing is and why it is so important.\nAfter reading this post, you will know => What natural language is and how it is different from other types of data.'

2. Cleaning

- Removing HTML Tag

In [51]:
```
def remove_html(text_data):
    """
    remove_html takes raw text and removes html tags from the text.
    """

    soup = BeautifulSoup(text_data, 'lxml')
    return soup.get_text()

processed_text = remove_html(str_data)
print(processed_text)
```

```
What is nlp???
Natural Language Processing, or NLP for short, is broadly defined as the
automatic manipulation of natural language, like speech and text, by sof
tware.
The study of natural language processing has been around for more than 5
0 years and grew out of the field of linguistics with the rise of comput
ers.
(In this post), you will discover what natural language processing is an
d why it is so important.
After reading this post, you will know => What natural language is and h
ow it is different from other types of data.
```

2. Cleaning

- Removing punctuation

```
## Check English's punctuation
print('Punctuation: ', string.punctuation)
```

```
Punctuation:  !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

```
def remove_punctuation(text):
    sent = []
    for t in text.split(' '):
        no_punct = "".join([c for c in t if c not in string.punctuation])
        sent.append(no_punct)

    sentence = " ".join(s for s in sent)
    return sentence
```

```
rmv_punc_sentence = remove_punctuation(processed_text)
print(rmv_punc_sentence)
```

```
What is nlp
Natural Language Processing or NLP for short is broadly defined as the automatic manipulati
on of natural language like speech and text by software
The study of natural language processing has been around for more than 50 years and grew ou
t of the field of linguistics with the rise of computers
In this post you will discover what natural language processing is and why it is so importa
nt
After reading this post you will know  What natural language is and how it is different fro
m other types of data
```

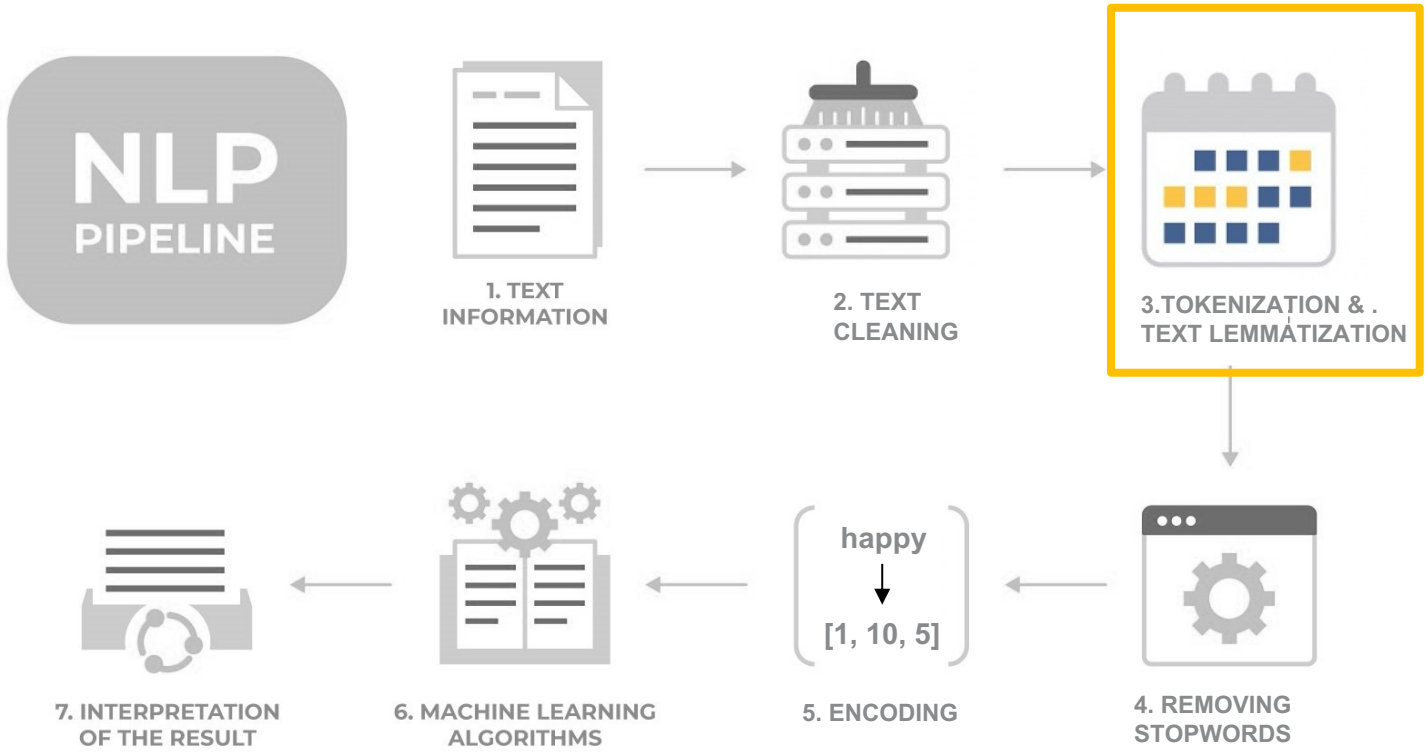
2. Cleaning

- Integration of upper and lower case letter

```
lower_sentence = rmv_punc_sentence.lower()
print(lower_sentence)
```

```
what is nlp
natural language processing or nlp for short is broadly defined as the automatic manipulati
on of natural language like speech and text by software
the study of natural language processing has been around for more than 50 years and grew ou
t of the field of linguistics with the rise of computers
in this post you will discover what natural language processing is and why it is so importa
nt
after reading this post you will know  what natural language is and how it is different fro
m other types of data
```

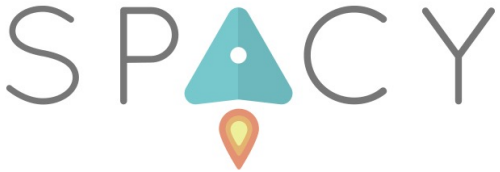
NLP preprocessing



NLP preprocessing

- What is spacy
  - a **free, open-source library** for advanced **Natural Language Processing** (NLP) in Python.

NAME
Tokenization
Part-of-speech (POS) **Tagging**
Dependency Parsing
Lemmatization
.....



3. Tokenization & Lemmatization

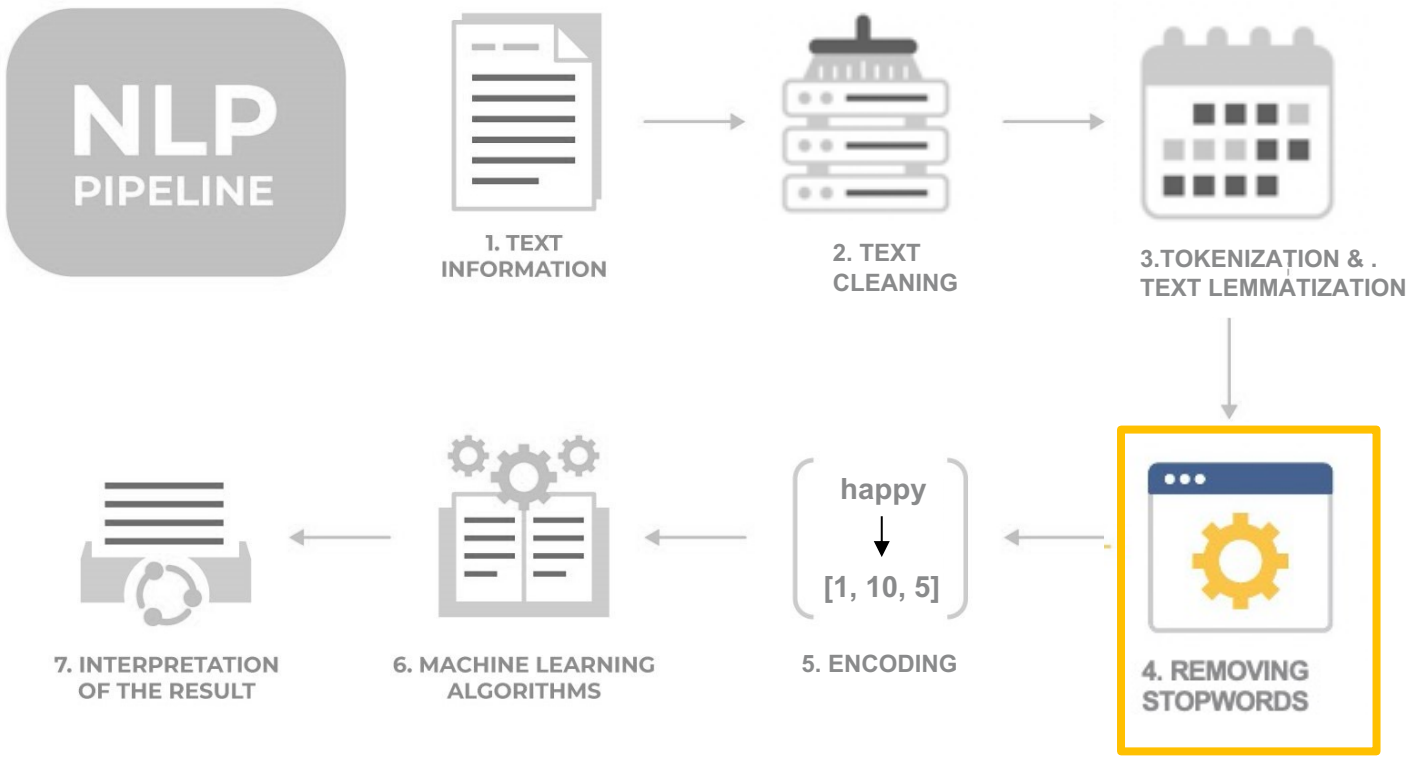
- **Tokenization(Sentence & Word)**
  - The process of tokenizing or splitting a string, text into a list of tokens
- **Lemmatization**
  - Transforming the word into a proper root form.
  - Considering the context and converting the word to its meaningful base form

'nlp' is installed model

```
In [104]: ## using "spacy" library
          import spacy

          ## Load the installed model "en_core_web_sm" into "nlp"
          nlp = spacy.load('en_core_web_sm')

In [105]: ## 'doc' is a sequence of Token objects
          ## it holds all information about the tokens, their linguistic features and their relationships.
          doc = nlp(lower_sentence.strip())

In [108]: tok_lem_sentence = [(token.text, token.lemma_) for token in doc]
          tok_lem_sentence[:15]

Out[108]: [('what', 'what'),
           ('is', 'be'),
           ('nlp', 'nlp'),
           (' \n', ' \n'),
           ('natural', 'natural'),
           ('language', 'language'),
           ('processing', 'processing'),
           ('or', 'or'),
           ('nlp', 'nlp'),
           ('for', 'for'),
           ('short', 'short'),
           ('is', 'be'),
           ('broadly', 'broadly'),
           ('defined', 'define'),
           ('as', 'as')]
```

token.text : tokenized
token.lemma_ : lemmatized

NLP preprocessing



4. Removing Stopwords

If No 'stopwords' exists
then run nltk.download('stopwords') first
to download them.

- Stopwords
  - a commonly used word such as 'the', 'a', 'an', 'in'.

In [107]:
```
from nltk.corpus import stopwords

print(stopwords.words('english')[:10])
print(len(stopwords.words('english')))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're"]
179
```

  - Removing stop words with NLTK

```
from nltk.corpus import stopwords

stop_words = set(stopwords.words('english'))

print(tok_lem_sentence, '\n')
rmv_sw_sentence = [w for w in tok_lem_sentence if not w in stop_words]
print(rmv_sw_sentence)
removed_word = [w for w in tok_lem_sentence if not w in rmv_sw_sentence]
print("\nRemoved word: ", set(removed_word))
```

4. Removing Stopwords

Before removing stopwords

```
['what', 'be', 'nlp', ' \n', 'natural', 'language', 'processing', 'or', 'nlp', 'for', 'short', 'be', 'broadly', 'defi
ne', 'as', 'the', 'automatic', 'manipulation', 'of', 'natural', 'language', 'like', 'speech', 'and', 'text', 'by', 's
oftware', '\n', 'the', 'study', 'of', 'natural', 'language', 'processing', 'have', 'be', 'around', 'for', 'more', 'th
an', '50', 'year', 'and', 'grow', 'out', 'of', 'the', 'field', 'of', 'linguistic', 'with', 'the', 'rise', 'of', 'comp
uter', '\n', 'in', 'this', 'post', 'you', 'will', 'discover', 'what', 'natural', 'language', 'processing', 'be', 'an
d', 'why', 'it', 'be', 'so', 'important', '\n', 'after', 'read', 'this', 'post', 'you', 'will', 'know', ' ', 'what',
'natural', 'language', 'be', 'and', 'how', 'it', 'be', 'different', 'from', 'other', 'type', 'of', 'datum']
```

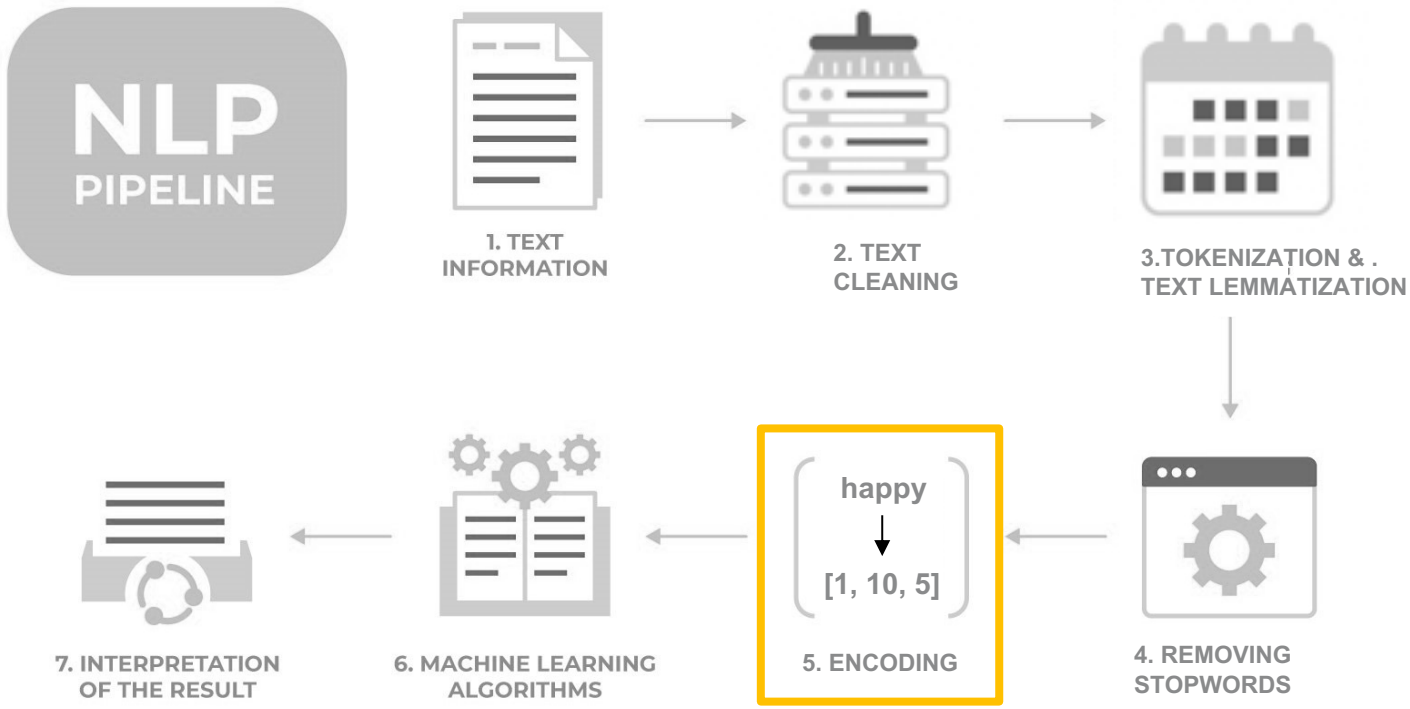
After removing stopwords

```
['nlp', ' \n', 'natural', 'language', 'processing', 'nlp', 'short', 'broadly', 'define', 'automatic', 'manipulation',
'natural', 'language', 'like', 'speech', 'text', 'software', '\n', 'study', 'natural', 'language', 'processing', 'aro
und', '50', 'year', 'grow', 'field', 'linguistic', 'rise', 'computer', '\n', 'post', 'discover', 'natural', 'languag
e', 'processing', 'important', '\n', 'read', 'post', 'know', ' ', 'natural', 'language', 'different', 'type', 'datu
m']
```

Removed stopwords

```
Removed word:  {'of', 'and', 'in', 'why', 'the', 'how', 'out', 'as', 'what', 'this', 'for', 'have', 'you', 'other',
'after', 'be', 'or', 'will', 'by', 'than', 'with', 'it', 'from', 'so', 'more'}
```

NLP preprocessing



5. Encoding

- Integer Encoding
  - What is integer encoding?
    - Mapping unique integers to words
  - How to?
    - Make a frequency-based dictionary
      - Step1. Make a frequency dictionary
      - Step2. Make a dictionary based on frequency
      - Step3. Add ‘OOV’ index for unknown words (OOV: Out of Vocabulary)
    - Encoding the words

5. Encoding – a frequency based dictionary

- Step1. Make a frequency dictionary

```
# save the data after removing stopwords
import numpy as np

dictionary = {}

def make_frequency_dict(text):
    for word in text:
        if word not in dictionary:
            dictionary[word] = 0
        dictionary[word] += 1

make_frequency_dict(rmv_sw_sentence)
```

```
len(dictionary)
```

```
33
```

```
dictionary
```

```
{'nlp': 2,
 ' \n': 1,
 'natural': 5,
 'language': 5,
 'processing': 3,
 'short': 1,
 'broadly': 1,
 'define': 1,
```

```
vocab_sorted = sorted(dictionary.items(), key=lambda x:x[1], reverse = True)
vocab_sorted
```

```
[('natural', 5),
 ('language', 5),
 ('processing', 3),
 ('\n', 3),
 ('nlp', 2),
 ('post', 2),
 (' \n', 1),
```

5. Encoding – a frequency based dictionary

- Step2. Make a dictionary based on frequency

```
word_to_index = {}
i = 0

for (word, frequency) in vocab_sorted :
    if frequency > 1 : # Cleaning: remove if frequency is less than 2
        i += 1
        word_to_index[word] = i
print(word_to_index)
```

```
{'natural': 1, 'language': 2, '\n': 3, 'processing': 4, 'nlp': 5, 'post': 6}
```

- Step3. Add 'OOV' index for unknown words

```
word_to_index['OOV'] = len(word_to_index) + 1
print(word_to_index)
```

```
{'natural': 1, 'language': 2, '\n': 3, 'processing': 4, 'nlp': 5, 'post': 6, 'OOV': 7}
```

5. Encoding – Encoding the words

```
encoded = []

print(rmv_sw_sentence)

for w in rmv_sw_sentence:
    encoded.append(word_to_index.get(w, word_to_index['OOV']))

print(encoded)
```

```
['nlp', ' \n', 'natural', 'language', 'processing', 'nlp', 'short', 'broadly', 'define', 'automatic', 'manipulation',
'natural', 'language', 'like', 'speech', 'text', 'software', '\n', 'study', 'natural', 'language', 'processing', 'aro
und', '50', 'year', 'grow', 'field', 'linguistic', 'rise', 'computer', '\n', 'post', 'discover', 'natural', 'languag
e', 'processing', 'important', '\n', 'read', 'post', 'know', '\n', 'natural', 'language', 'different', 'type', 'datu
m']
[5, 7, 1, 2, 4, 5, 7, 7, 7, 7, 7, 1, 2, 7, 7, 7, 7, 3, 7, 1, 2, 4, 7, 7, 7, 7, 7, 7, 7, 7, 3, 6, 7, 1, 2, 4, 7, 3, 7,
6, 7, 3, 1, 2, 7, 7, 7]
```

Exercise (2)

- complete Practice02_NLP code following the slides.

5. Encoding

- One-hot encoding
  - Problem in Integer encoding
    - Giving the higher numbers higher weights.
  - What is one-hot encoding?
    - A method to quantify categorical data
    - Producing a vector with length equal to the number of categories in the data set.

Label Encoding

Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

index

→

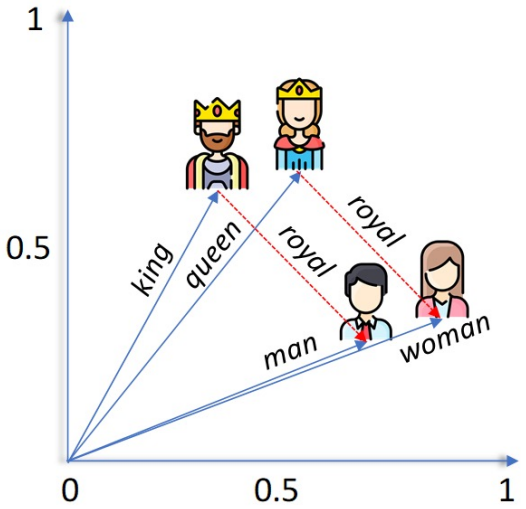
One Hot Encoding

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

vector

Word Embedding

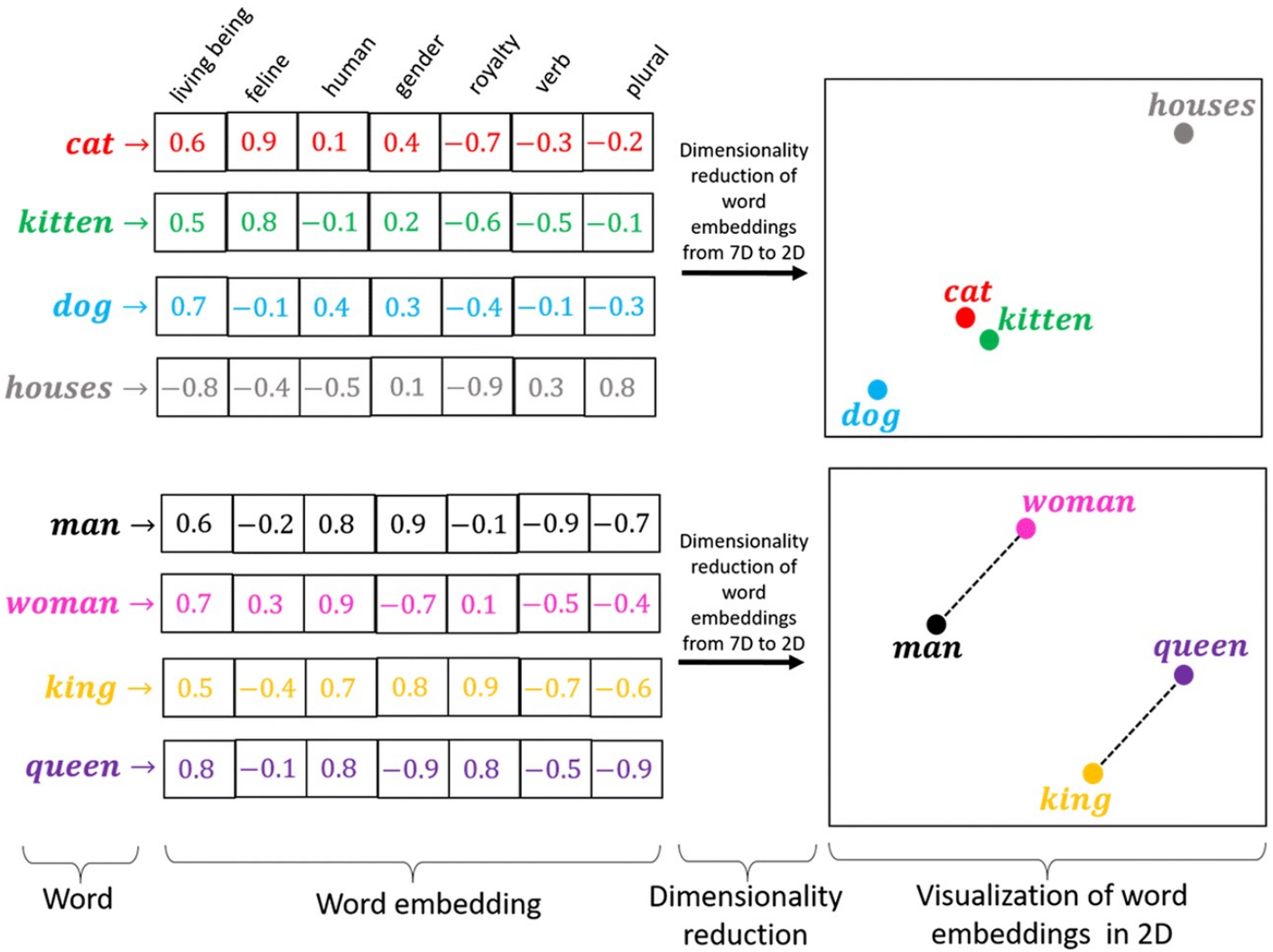
- Word Embedding
  - Problem in One-hot encoding
    - Vector length = the number of words in the dictionary
    - Losing the context of the sentence
  - What is word embedding?
    - Vector representations of a particular word
    - Capturing context of a word in a document, semantic and syntactic similarity, relation with other words
  - Word embedding type
    - Frequency based embedding
    - Prediction based embedding
      - *Example : Word2Vec*



https://towardsdatascience.com/deep-learning-for-nlp-word-embeddings-4f5c90bcdab5

NLP preprocessing

- Word Embedding



https://medium.com/@hari4om/word-embedding-d816f643140

Exercise (3)

	review_title	rating	review_date	customer_name	review
0	One Star	1.0	25 July 2014	By\n \n Andrea Bradden\n \n on 25 July...	ordered this, there was no PB embroidered on ...
1	Arm missing!!	1.0	1 Nov. 2015	By\n \n gemma james\n \n on 1 Nov. 2015	These are smaller than than you think and a l...
2	Cheap advent calendar	1.0	28 Oct. 2015	By\n \n lully\n \n on 28 Oct. 2015	Thought this would make a lovely different ca...
3	Poor quality sand	1.0	26 Dec. 2015	By\n \n Amazon Customer\n \n on 26 Dec...	The sand is rubbish - very messy and doesn't ...
4	Colour choice	1.0	19 Dec. 2015	By\n \n Pen Name\n \n on 19 Dec. 2015	Know it says random colours but wish we could...
...	...	...	...	...	...
495	Five Stars	5.0	29 Sept. 2014	By\n \n D. G. Long\n \n on 29 Sept. 2014	My daughter loves this and runs and jumps abo...
496	Five Stars	5.0	5 Jan. 2016	By\n \n Paul Cavanagh\n \n on 5 Jan. 2...	Great model
497	Fantastic detail! A beautiful model traction e...	5.0	23 Nov. 2015	By\n \n JET\n \n on 23 Nov. 2015	Fantastic detail! A beautiful model traction ...
498	very good quality	5.0	7 July 2013	By\n \n Storm\n \n on 7 July 2013	easy to couple with other models, great to ex...
499	Excellent	5.0	30 April 2011	By\n \n Ella\n \n on 30 April 2011	I bought this for my 2 year old grandson and ...

500 rows × 5 columns

Exercise (3)

- print the most 5 frequent words for each review value from ’amazon_train_df.csv’

- output format.

  review 1: Z, K, C, D, E

  review 2: S, F, G, B, M

  review 3: …

  review 4: …

  review 5: T, X, P, Z, K