Intro to text mining - Text processing - 1

One should look for what is and not what he thinks should be. (Albert Einstein)

Text Processing: Topic introduction

In this part of the course, we will cover the following concepts:

- The need for text processing and the tools used to perform it
- Definition and implementation of text processing steps
- Word distribution in a corpus

Video

- Before starting the next module, let’s watch a video on text mining:
- ***https://www.youtube.com/watch?v=xxqrIZyKKuk***

Module completion checklist

Objective	Complete
Review the tools and packages in Python to work with text data; create and inspect a corpus object	
Discuss the specific steps to pre-process text for the bag-of-words approach	

The need of Text processing in NLP

- **Text processing: A catalyst for model efficiency**
  - **Enhances model accuracy**: By stripping away irrelevant characters like punctuation, emojis, and common filler words, text preprocessing sharpens the focus on meaningful data, directly impacting the model's ability to discern and learn
  - **Optimizes data utility**: Particularly in handling user-generated content (e.g., tweets), it isolates the core sentiment or intent by distilling expressions to their base form (e.g., transforming "looooving" to "love"), ensuring the model trains on data that truly matters
  - **Streamlines computational resources**: Reduces the model's processing load by eliminating extraneous elements, thereby speeding up analysis and improving resource efficiency—crucial for scalability and real-time processing needs

Load packages

- We have used the helper packages so far, the packages that are new are all that are related to text processing.

```
# Helper packages.
import pandas as pd
import numpy as np
import warnings
import matplotlib.pyplot as plt
from pathlib import Path
```

```
# Packages with tools for text processing.
from wordcloud import WordCloud
import nltk
import nltk.data
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer
from nltk.util import ngrams

# Get common English stop words.
stop_words = stopwords.words('english')
```

Directory settings

- In order to maximize the efficiency of your workflow, you should encode your directory structure into `variables`
- We will use the `pathlib` library
- Let the `main_dir` be the variable corresponding to your `course` folder
- Let `data_dir` be the variable corresponding to your `data` folder

```
# Set 'main_dir' to location of the project folder
home_dir = Path(".").resolve()
main_dir = home_dir.parent.parent
print(main_dir)
```

```
data_dir = str(main_dir) + "/data"
print(data_dir)
```

A little about NLTK

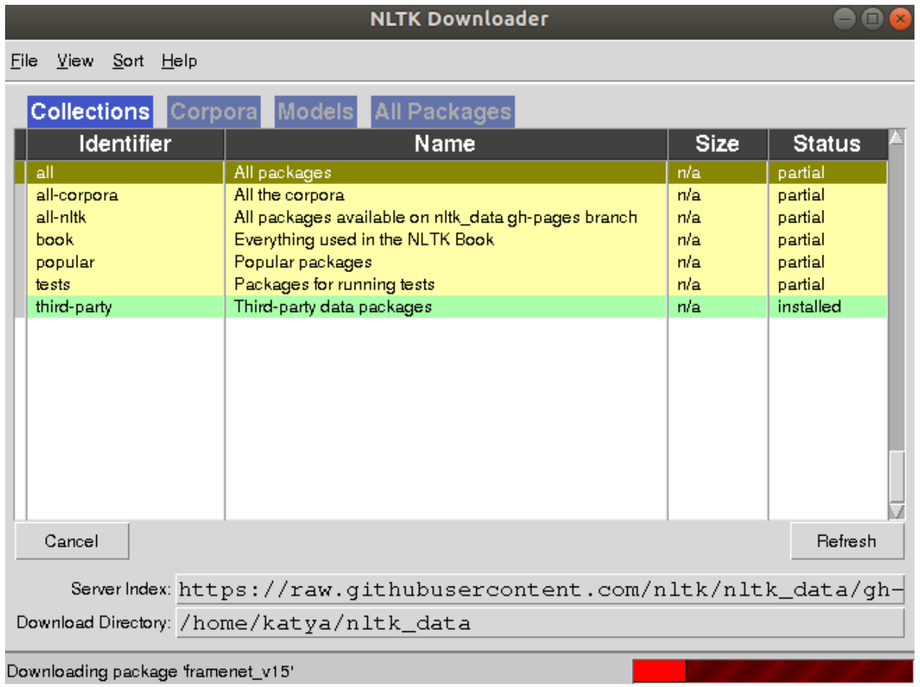
- We will be using NLTK to dive into NLP and text analysis
- Python's **Natural Language Tool Kit**, or **NLTK** as it is known, is a very powerful platform for teaching, and working in, **computational linguistics using Python**
- It is free, open source, easy to use, large community and ***well documented***

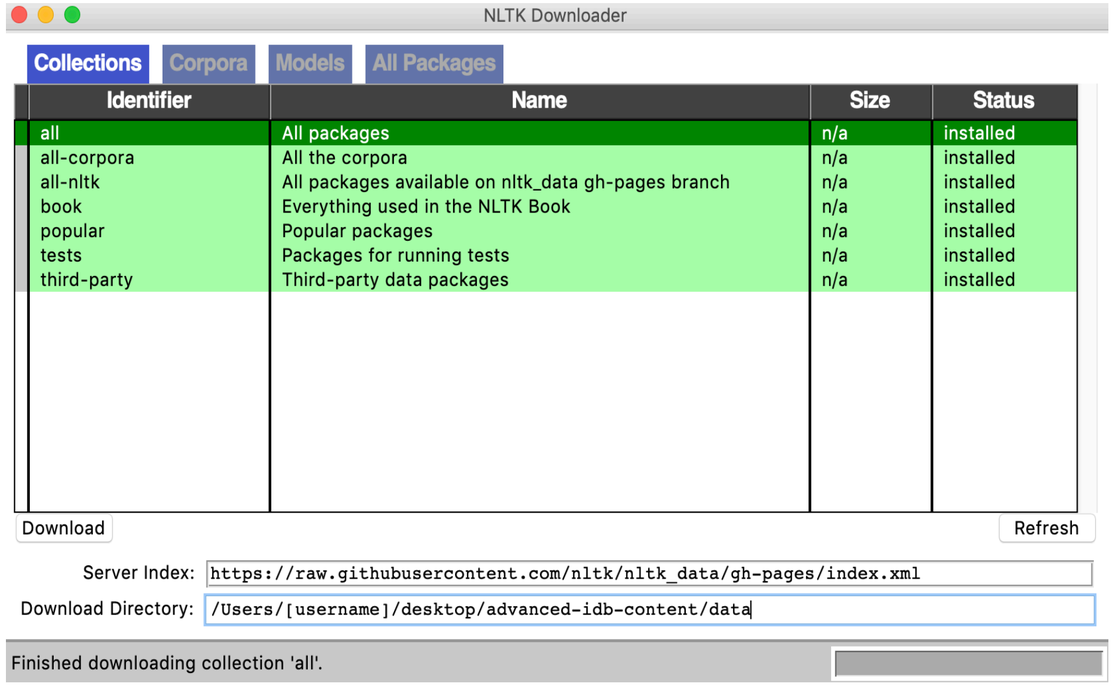
Download NLTK resources

```
# Command to download resources from NLTK
package.

#nltk.download()

# Uncomment above command to download,
# Once successfully downloaded, comment it back
to avoid repetitive download trials
```





- When you have downloaded all resources, just close this dialogue window

New York Times articles: case study

- The New York Times is a widely read American newspaper with an extensive textual archive
- We can better understand its impact by analyzing its frequently used terms and identifying the topics discussed in each article
- Our starting point is a dataset from *Kaggle* containing **snippets** of the paper's articles
- We will analyze these text snippets to find the **most frequent terms**
- We will also extract article topics and **tag** each article with a topic to better understand the main subjects of the articles

Dataset

- In order to implement what you learn in this course, we will be using the NYT_article_data.csv dataset
- We will be working with columns from the dataset such as:
  - `web_url`
  - `headline`
  - `snippet`
  - `word_count`
  - `source`

Load text data

- We will now load the dataset and save it as `df`

```
df = pd.read_csv(str(data_dir)+"/"+ "NYT_article_data.csv")
print(df.columns)
```

```
Index(['web_url', 'headline', 'snippet', 'word_count', 'source',
       'type_of_material', 'date', 'id'],
      dtype='object')
```

- We will be focusing on the `snippet` column as of now

```
# Look at the columns.
print(df["snippet"].head())
```

```
0    Nick Kyrgios started his Brisbane Open title d...
1    British police confirmed on Tuesday they were ...
2    Marcellus Wiley is still on the fence about le...
3    Still reckoning with the fallout from her Emme...
4    As far as Arike Ogunbowale and coach Muffet Mc...
Name: snippet, dtype: object
```

Drop missing data

- Check and drop any NAs or empty values from the `snippet` column as they are not relevant for our analysis

```
# Print total number of NAs.
print(df["snippet"].isna().sum())
```

```
0
```

```
# Drop NAs if any.
df = df.dropna(subset = ["snippet"]).reset_index(drop=True)
print(df["snippet"].isna().sum())
```

```
0
```

Units of text

- In this case, we will consider:
  - Each entry in **snippet** as a **document**
  - **All entries in snippet** together as a **corpus**
- This division of units of text will help us:
  - Evaluate each entry in `snippet` for its subject and potential topic
  - Evaluate all entries in `snippet` for general word distribution and overall patterns

Create a series of documents

```
# Isolate the `snippet` column.
df_text = df["snippet"]
print(type(df_text))
```

```
<class 'pandas.core.series.Series'>
```

- Look at a sample of the documents:

```
# Look at a sample of the `snippet` column, 0-5.
print(df_text[0:5])
```

```
0    Nick Kyrgios started his Brisbane Open title d...
1    British police confirmed on Tuesday they were ...
2    Marcellus Wiley is still on the fence about le...
3    Still reckoning with the fallout from her Emme...
4    As far as Arike Ogunbowale and coach Muffet Mc...
Name: snippet, dtype: object
```

Module completion checklist

Objective	Complete
Review the tools and packages in Python to work with text data; create and inspect a corpus object	✔
Discuss the specific steps to pre-process text for the bag-of-words approach	

Where do we go from here?

- Once the text data is loaded, we need to move to the next stage: **text data preparation**
- Why do we need to prepare the data?
- How do we go about it?

“Bag-of-words” analysis

- How do we quantify and compute on text data?
  - *We need to translate words into numbers*
- How do we translate words into numbers?
  - *The simplest solution is to* ***count*** *them*
- The analysis of text data based on word counts (a.k.a. frequencies) in documents is called **bag-of-words**

Index	Word	Freq	%
A	Apple	5	20
B	Book	7	28
C	Cat	13	52

“Bag-of-words” analysis: from text to numbers

1. **A document is** treated as **a bag of words** where word position and structure do not matter
2. Text is cleaned until only **stripped down word-roots** remain
3. **Each occurrence** of a word is then counted in each document
4. **Word frequencies** are recorded and arranged into a matrix of words by documents, additional weighting may be applied
5. The **numeric representation** of your text corpus is this **matrix**
6. Document similarity is based on the **similar words** that appear in **documents** with similar **meaning**

This is the most basic version of the bag of words data preparation flow!

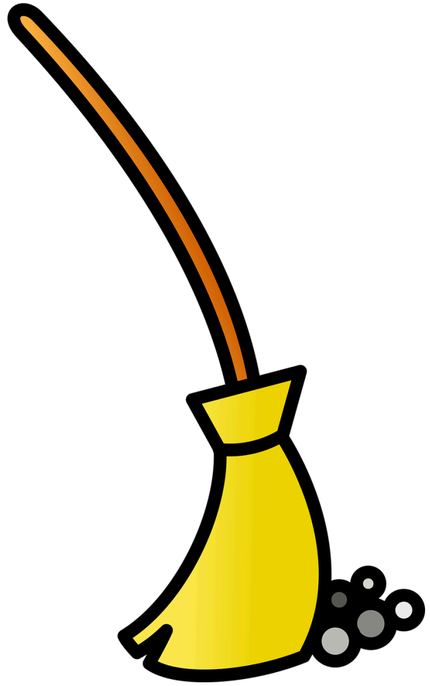
“Bag-of-words” analysis: key elements

What we need	What we have learned
A corpus of documents cleaned and processed in a certain way • All words a converted to lowercase • All punctuation, numbers and special characters are removed • Stopwords are removed • Words are stemmed to their root form (and sometimes lemmatized)	
A Document-Term Matrix (DTM): with counts of each word recorded for each document	
A transformed representation of a Document-Term Matrix (i.e. weighted with TF-IDF weights)	

“Bag-of-words” analysis: cleaning text flow

Text preparation and cleaning is one of the most important steps in text mining and analysis

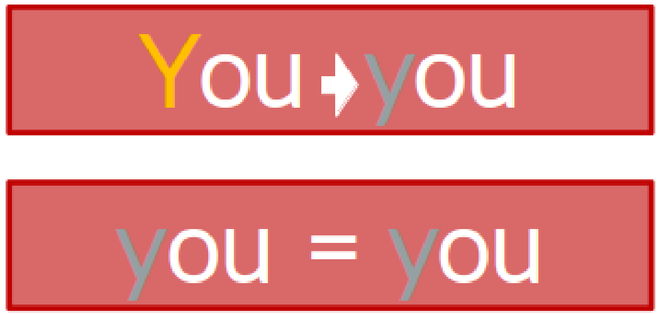
1. Convert all characters to lowercase
2. Remove stop words
3. Remove punctuation, numbers, and all other symbols that are not letters of the alphabet
4. Stem words
5. Remove extra white space (if needed)



Text cleaning steps: order does matter!

dos

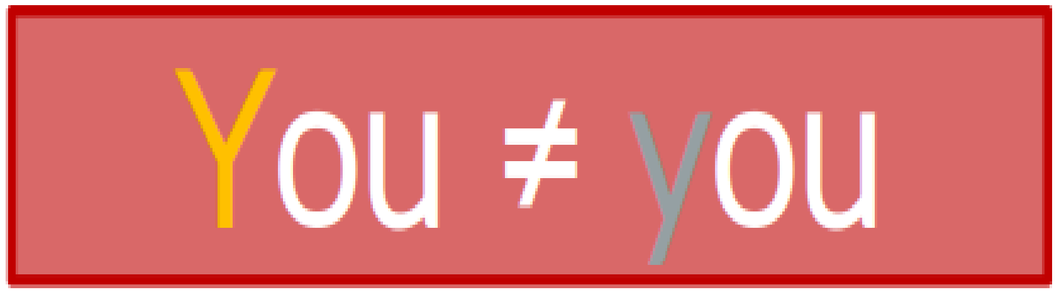
1. Convert all characters to lowercase
2. Remove stop words



- When we first unify all words and convert them to lowercase, we will be able to catch all instances of stop words!

don’ts

1. Remove stop words
2. Convert all characters to lowercase



- All stop word dictionaries are in all lowercase. If we do not convert our text to lowercase, those stop words that were in upper case will be ignored!

Knowledge check

Module completion checklist

Objective	Complete
Review the tools and packages in Python to work with text data and create and inspect a corpus object	✔
Discuss the specific steps to pre-process text for the bag-of-words approach	✔

Congratulations on completing this module!

You are now ready to try Tasks 1-3 in the Exercise for this topic