Tutorial Pack 12
(90 minutes)

(To be completed during LW12 tutorial)

LEARNING OBJECTIVES

- **To practice working with data collected from the YouTube Data API**
- **To practice applying sentiment scoring models**

LEARNING OUTCOMES

- **By the end of this tutorial students will have;**
  - **Read in a sample dataset in TSV format into a dataframe.**
  - **Familiarised themselves with common sentiment lexicons.**
  - **Extracted fields from a data file in JSON format.**
  - **Carried out filtering on a Pandas dataframe.**
  - **Applied the VADER model to a text dataset.**

RESOUCRES AND TOOLS REQUIRED

- **Lecture Slides for LW 11**
- **Google Collab Notebook**
- **Data files: 2000.tsv, yt[1-5].json**

IMPORTANT:

This pack is designed for you to go at your own speed. At the end of each section there is a series of practice questions and exercises. You should attempt to answer **all questions**.

Any questions you do not complete today should be completed before your next tutorial.

You may find this tutorial pack and the exercises useful preparation for the coursework.

1. Sentiment Lexicons

Recall from the week 11 lecture, a sentiment lexicon is a word or phrase that has been labelled with a sentiment **orientation**.

In this tutorial, we will make use of a pre-labelled set of sentiment lexicons ("2000.tsv") based on the top ~5000 most frequently used words between 1990-2000. This extract is part of the SocialSent Sentiment Data.

The sentiment lexicon data can be read in using the *read_csv* function. Since the file contains data separated by tabs, we specify "\t" as the separation parameter.

```
import pandas as pd

df_lex = pd.read_csv("2000.tsv", sep="\t", header=None)
df_lex.columns=["word","sentiment", "std.dev"]
```

Figure 1 Reading in the sample lexicons

The dataframe consists of three columns, the word, the normalised sentiment score, and the standard deviation of the sentiment score.

	word	sentiment	std.dev
0	ugly	-3.90	1.16
1	painful	-3.69	1.53
2	intent	-3.49	1.67
3	terrible	-3.38	1.55
4	drunk	-3.28	1.16
...	...	...	...
4919	perfectly	2.69	0.83
4920	romantic	2.70	0.76
4921	delicate	2.72	0.93
4922	beautiful	2.73	0.69
4923	wonderful	2.76	0.71

Figure 2 Dataframe of sample lexicons

2. Super Bowl 2023 Commercial

The Super Bowl is the annual final playoff game of the National Football League (NFL) to determine the league champion. Due to the number of people that watch the event, it has become extremely popular with advertisers.

In 2023, one of the commercials shown during the event was for the brand PopCorners. This commercial was also posted to YouTube and has attracted more than 5 million views and many thousands of comments.

Comments posted to this video have been extracted using the YouTube Data API and placed into a series of text files (named yt[1-5].json) in the **JSON** format.


```
{
  "kind": "youtube#commentThreadListResponse",
  "etag": "EwHGiUbK_tDxJfPxa__d5E5aSLs",
  "nextPageToken": "QURTS19pM09mRldLNERzcDQxNk50YjdubTN0MWZRcmNvZnpMZHBBUmRjekJRZnJ1MTY1X1pLdzdPbDkyc2JCTVgxcFZYdl81cEZvRk5xcw==",
  "pageInfo": {
    "totalResults": 100,
    "resultsPerPage": 100
  },
  "items": [
    {
      "kind": "youtube#commentThread",
      "etag": "MZ1RFbZmwxErmjo2NA5k-KBGOeU",
      "id": "Ugz34z9rtEtJcsTp4yp4AaABAg",
      "snippet": {
        "videoId": "ZMlemd6U24Y",
        "topLevelComment": {
          "kind": "youtube#comment",
          "etag": "QAj0m07jrBxJxKACiPOAz0Jk154",
          "id": "Ugz34z9rtEtJcsTp4yp4AaABAg",
          "snippet": {
            "videoId": "ZMlemd6U24Y",
            "textDisplay": "They had to smash it with the handle of a knife before eating it.",
            "textOriginal": "They had to smash it with the handle of a knife before eating it.",
            "authorDisplayName": "Joachim männlich • 82 years ago",
            "authorProfileImageUrl": "https://yt3.ggpht.com/dld0PFZ4BX5p4bZAUglBD7Jj41bCQfwkVDamuJBNGSiesrpLb57uuXKCevvpq_7eV-HLpwVjX
            "authorChannelUrl": "http://www.youtube.com/channel/UC7nIB9JGFVcVXH37cal41EA",
            "authorChannelId": {
              "value": "UC7nIB9JGFVcVXH37cal41EA"
            },
            "canRate": true,
            "viewerRating": "none",
            "likeCount": 0,
            "publishedAt": "2023-04-09T13:53:03Z",
            "updatedAt": "2023-04-09T13:53:03Z"
          }
        },
        "canReply": true,
        "totalReplyCount": 0,
        "isPublic": true
      }
    },
```


Figure 3 Example comment data from the YouTube Data API

The Python package **json** can be used to read JSON files. To extract the comments from the five files we use the following methodology.

1. Open each JSON file sequentially starting at yt1.json and ending at yt5.json
2. Load the contents of each file into a Python json data structure.
3. Loop over the array of “items” stored in each file, where an item represents an individual comment.
4. Access the “textDisplay” property of each item by walking down the data dictionaries associated with each “item”.
5. Populate a list of comments based on the “textDisplay” property.
6. Finally, create a new dataframe based on the complete list of comments.

The code that implements this methodology is shown below. It is also attached at the end of this tutorial pack in the appendix.

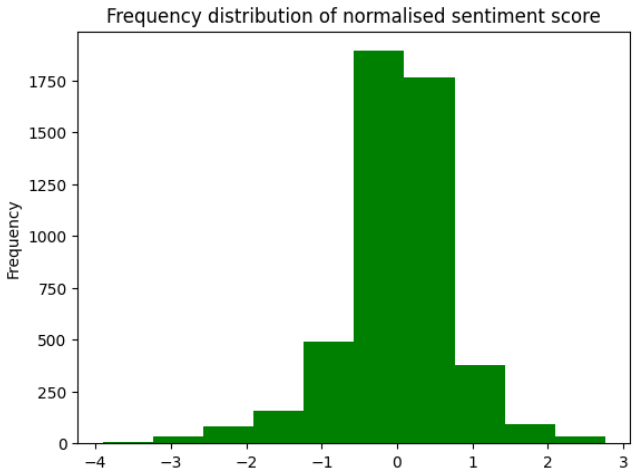
```
import json

comments = []
for i in range(1,6):
  with open("yt%i.json" % i, "r") as f:
    data = json.load(f)
    for item in data["items"]:
      comment = item["snippet"]["topLevelComment"]["snippet"]["textDisplay"]
      comment = comment.strip()
      if len(comment) < 10:
        continue
      comments.append({"text": comment})
df = pd.DataFrame(comments)
```

Figure 4 Code used to extract the comments from a set of JSON files (YouTube Data API)

Questions

Q1 Import the sample lexicon data into a data frame called “**df_lex**”. Create a histogram of the sentiment scores of all words and comment of whether or not the scores appear to follow a normal distribution.

Your output should resemble the screenshot below.



Q2 What is the average sentment score across all words in the sample dataset? Verify this with a suitable Python statement.

Q3 Calculate the average sentiment score of the top 5 most positive lexicons. Your answer should be rounded to 2dp. **Hint**: use the round() method.

Q4 Assuming a negative word has a sentiment score of less than -0.5, how many lexicons in the sample dataset fall into the negative category? **Hint:** use a filter condition.

Q5 Populate dictionary called “mapping” whose keys and values correspond to the words and sentiment scores respectively in the same dataset. Your dictionary should resemble the following. **Hint**: Use the to_dict(“records”) method of your lexicon dataframe.

```
{'ugly': -3.9,
 'painful': -3.69,
 'intent': -3.49,
 'terrible': -3.38,
 'drunk': -3.28,
```

Q6 Create a function called *sentiment_score*() that calculates a simple setiment score based on the average sentiment of words in a comment. Your function should make use of the mapping dictionary created in Q5 and accept a single string as a parameter.

Use the following comment to test your function words as expected:

"i think breaking bad is boring!" => -0.68 sentiment score

Q7 Using the code in figure 4, import the youtube comments into a new dataframe and use a modified version of the function created in Q6 to calculate a new column called "simple_sentiment".

Your datafame should resemble the screenshot below.

	text	simple_sentiment
0	They had to smash it with the handle of a knif...	-0.303333
1	YEAH SIMPLE INGREDIENTS MR BETCH	0.550000
2	Tuco's looking pretty tight yo ❤	0.763333
3	I'm wondering if these are keto friendly.. Lol.	-0.860000
4	They ain't that good low key	0.305000
...	...	...
462	This is literally the reason why i bought popc...	-0.061667
463	aint no way they actually made this a commerci...	-0.166667
464	When I tasted the white cheddar popcorners shi...	0.030000
465	Imagine the production and recipe teams for Po...	0.112727
466	I seen the commercial on Super Bowl LVII and g...	-0.278333

Q8 Consider the comment "*Saw the commercial I tried them as a joke then I absolutely loved them*" Why might analysing this comment word by word may understate its true sentiment?

Q9 Consider the following Python code. What do you expect the following code to display when run and explain its purpose,

```
comment = "Saw the commercial I tried them as a joke then I absolutely loved them"
words = comment.lower().split()
for left, right in zip(words[:-1], words[1:]):
  print(left, right)
```

Q10 Add a new column to your dataframmed called “vader_sentiment”. Using the code shown in LW11 lecture slides, calculate the sentiment for each comment based on vader’s compound score.

Your output should resemble the screenshot below.

	text	simple_sentiment	vader_sentiment
0	They had to smash it with the handle of a knif...	-0.303333	0.0000
1	YEAH SIMPLE INGREDIENTS MR BETCH	0.550000	0.2960
2	Tuco’s looking pretty tight yo ♥	0.763333	0.4939
3	I’m wondering if these are keto friendly.. Lol.	-0.860000	0.4215
4	They ain’t that good low key	0.305000	0.2023
...	...	...	...
462	This is literally the reason why i bought popc...	-0.061667	-0.2960
463	aint no way they actually made this a commerci...	-0.166667	0.2235
464	When I tasted the white cheddar popcorners shi...	0.030000	-0.5574
465	Imagine the production and recipe teams for Po...	0.112727	0.0000
466	I seen the commercial on Super Bowl LVII and g...	-0.278333	0.8449

467 rows × 3 columns

Q11 Graph the sentiment score calculated for each approach and inspect the sentiment distribution. Comment on the extent to which the commercial is polarising.

Appendix 1

```
import json

comments = []
for i in range(1,6):
  with open("yt%i.json" % i, "r") as f:
    data = json.load(f)
    for item in data["items"]:
      comment = item["snippet"]["topLevelComment"]["snippet"]["textDisp
lay"]
      comment = comment.strip()
      if len(comment) < 10:
        continue
      comments.append({"text": comment})
df = pd.DataFrame(comments)
```