lab_session2

May 1, 2022

1 Lab session #2

1.1 Data processing, Statistics and Visualisation, Clustering

Please enter your firstname and lastname below.

- Firstname:
- Lastname:

In this assignment, the input is a csv file containing ELI5 posts (question, topics and other miscelleanous information) which we convert for you into a Pandas dataframe of the form:

	id	answer	score	topic
0	0	Whoever said that is wrong.The FDA and IWBA ca...	14100.0	Chemistry
1	1	In Europe you have the time of bottling printe...	1400.0	Chemistry
2	2	I'd hate to be "that guy" but I was looking an...	47.0	Chemistry
3	3	I've heard of several studies that conclude th...	195.0	Chemistry
4	4	UV and heat will degrade the material however.	15.0	Chemistry

The assignment involves processing the text, exploratory analysis of the data (statistics and visualisation) and clustering.

1.2 1 - Data pre-processing (14 points, 2 Bonus points)

IMPORTANT Make sure to run the following cell before starting on the exercises.

[ ]:
```
import pandas as pd
df = pd.read_csv("eli5questions.csv", sep = ",",on_bad_lines='skip')
df.head()
```

Exercice 1.1 (2 points)

How many topics and how many instance per topic ? - Extract from the dataframe the list of possible topics and the number of instances (rows) per topic.

Hint Cf. Pandas cheat sheet

[ ]:
```
# How many topics and how many instance per topic ?

# YOUR CODE HERE
```

Exercise 1.2 (6 points)

Define a function "preprocessing(text)" which takes as input a string and returns a new, modified string which results from:

- tokenizing the input string
- lower casing the resulting tokens
- removing all tokens that are not made of alphabetical characters (use python isalpha method)
- converting the resulting list of tokens back into a string (use python join method)

[ ]:
```
# YOUR CODE HERE
```

Exercise 1.3 (2 points)

- Apply the preprocessing function to the "answer" column of the df dataframe provided above and store the result into a new dataframe called "clean_answers".
- print out the first 5 rows of this new dataframe

Hint: Use Pandas "apply" and "head" methods

[ ]:
```
# YOUR CODE HERE
```

Exercice 1.4 - how much does cleaning reduce the size of the input (2 points) ?

- Compute the number of unique tokens (vocabulary) contained in (i) the "answer" column of the initial dataframe and (ii) the clean_answer dataframe you just created. As the string is very large, feel free to use split to tokenize (rather than a tokenizer)
- Print out both numbers

[ ]:
```
# YOUR CODE HERE
```

Exercise 1.5 (OPTIONAL, 2 BONUS POINTS)

- Define a function "get_content_words(text)" which takes as input a string and returns the nouns and verbs it contains as a string
- Apply this function to the "answer" column of the df dataframe and store the result into a new dataframe called "cw_df".
- Compute the number of unique tokens contained in (i) the "answer" column of the initial dataframe df and (ii) the content words column of the cw_df dataframe you just created
- Print out both numbers

Hints - the spacy pos tags for verbs and nouns are 'VERB' and 'NOUN' respectively. - str(x) converts x to a string (you might need this when using spacy)

[ ]:
```
# YOUR CODE HERE
```

Exercise 1.6: Create training data for the clustering exercise (2 points)

- Extract the topic column of the df dataframe
- Create a new dataframe called "data_df" containing columns for
- the cleaned up answers
- the topics of each answer
- content words (if you have computed them for the BONUS point questions)

Hint: Use pandas concat method

The output should be something like this if you did the bonus questions, else it will only contain the clean answers and the topic column:

	clean answers	content words	topic
0	whoever said that is wrong the fda and iwba ca...	Whoever said is ca find evidence age matters p...	Chemistry
1	in europe you have the time of bottling printe...	have time bottling printed bottle way can figu...	Chemistry
2	i hate to be that guy but i was looking and no...	'd hate be guy was looking one has chimed boug...	Chemistry
3	i heard of several studies that conclude that ...	've heard studies conclude plastic bottles do ...	Chemistry
4	uv and heat will degrade the material however	UV heat will degrade material	Chemistry

[ ]:
```
# YOUR CODE HERE
```

1.3 2 - Statistics and Visualisation (6 points)

Exercise 2.1 (4 points)

- Define a function "tokenize_and_count" which takes as input a string, tokenizes it and returns the number of tokens produced.
- Apply this function to the "answer" content of the data_df dataframe created in Exercise 1.3 and update data_df with the results.
  data_df should now contain 3 columns with headers "answer", "topic" and "nb_tokens".

The output should be something like this (or only the columns clean answers, topic and nb_tokens if did not do the bonus questions):

	clean answers	content words	topic	nb_tokens	nb_cw
0	whoever said that is wrong the fda and iwba ca...	Whoever said is ca find evidence age matters p...	Chemistry	142	73
1	in europe you have the time of bottling printe...	have time bottling printed bottle way can figu...	Chemistry	28	11
2	i hate to be that guy but i was looking and no...	'd hate be guy was looking one has chimed boug...	Chemistry	46	23
3	i heard of several studies that conclude that ...	've heard studies conclude plastic bottles do ...	Chemistry	58	30
4	uv and heat will degrade the material however	UV heat will degrade material	Chemistry	8	5

[ ]:
```
# YOUR CODE HERE
```

Exercise 2.2 (2 points) Plot the histogram of number of tokens

[ ]:
```
# YOUR CODE HERE
```

1.4 3 - Clustering (10 points)

Exercise 3.1 (2 points) Use sklearn TfidfVectorizer method to turn the clean answers (or the content words) into a TF-IDF matrix where each row represents an answer, the columns are tokens and the cell contains the tf-idf score of each token.

- Import the TfidfVectorizer method from sklearn
- Create a tf-idf vectorizer. The maximum nb of features should be set to 8000. Set use_idf to True, stop_words to "english" and the tokenizer to nltk.word_tokenize.
- set
- Apply the tfidf_vectorizer.fit_transform method to the clean answers (extract these from the data_df dataframe created in Exercise 1.6)

[ ]:
```
# YOUR CODE HERE
```

Exercise 3.2 Training a K-means clustering model (2 points)

- Create a K-means object (import KMeans from sklearn.cluster)
- train this object on the tf-idf matrix you created from the data (use the kmeans fit method)

[ ]:
```
# YOUR CODE HERE
```

Exercise 3.3 Print out the top terms of each clusters (4 points)

[ ]:
```
# YOUR CODE HERE
```

Exercise 3.4 Evaluate the clusters (2 points)

- Import metrics from sklearns and compute homogeneity, completeness, v_measure, adjusted rand index and silhouette coefficient
- Print each score out

Hint: the metrics methods (homogeneity_score, completeness_score etc.) take as input the list of true labels (Y) and the list of predicted labels which you can get using kmeans labels_ attribute on the clustering model you have learned in the previous exercise (e.g., km.labels_ if you've called your model km).

[ ]:
```
# YOUR CODE HERE
```

Visualising the clusters (PROVIDED) You might need to adjust the variables to fit your code

[ ]:
```
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.manifold import MDS
```

```
dist = 1 - cosine_similarity(tfidf_matrix)

# Use multidimensional scaling to convert the dist matrix into a 2-dimensional␣
 ↪array
MDS()

# n_components=2 to plot results in a two-dimensional plane
mds = MDS(n_components=2, dissimilarity="precomputed", random_state=1)
pos = mds.fit_transform(dist)
xs, ys = pos[:, 0], pos[:, 1]

#set up colors per clusters
cluster_colors = {0: '#1b9e77', 1: '#d95f02', 2: '#7570b3'}

#set up cluster names
cluster_names = {0: 'Other', 1: 'Chemistry', 2: 'Physics'}

#create data frame that has the result of the MDS plus the cluster numbers and␣
 ↪titles
df = pd.DataFrame(dict(x=xs, y=ys, label=km.labels_.tolist()))

#group by cluster
groups = df.groupby('label')

# set up plot
fig, ax = plt.subplots(figsize=(17, 9))
ax.margins(0.05)

#iterate through groups to layer the plot
for name, group in groups:
    ax.plot(group.x, group.y, marker='o', linestyle='', ms=12,
            label=cluster_names[name],
            color=cluster_colors[name],
            mec='none')
    ax.set_aspect('auto')
    ax.tick_params(\
        axis= 'x',
        which='both',
        bottom=False,
        top=False,
        labelbottom=False)
    ax.tick_params(\
        axis= 'y',
        which='both',
        left=False,
        top=False,
        labelleft=False)
```

```
ax.legend(numpoints=1)
plt.show()
```