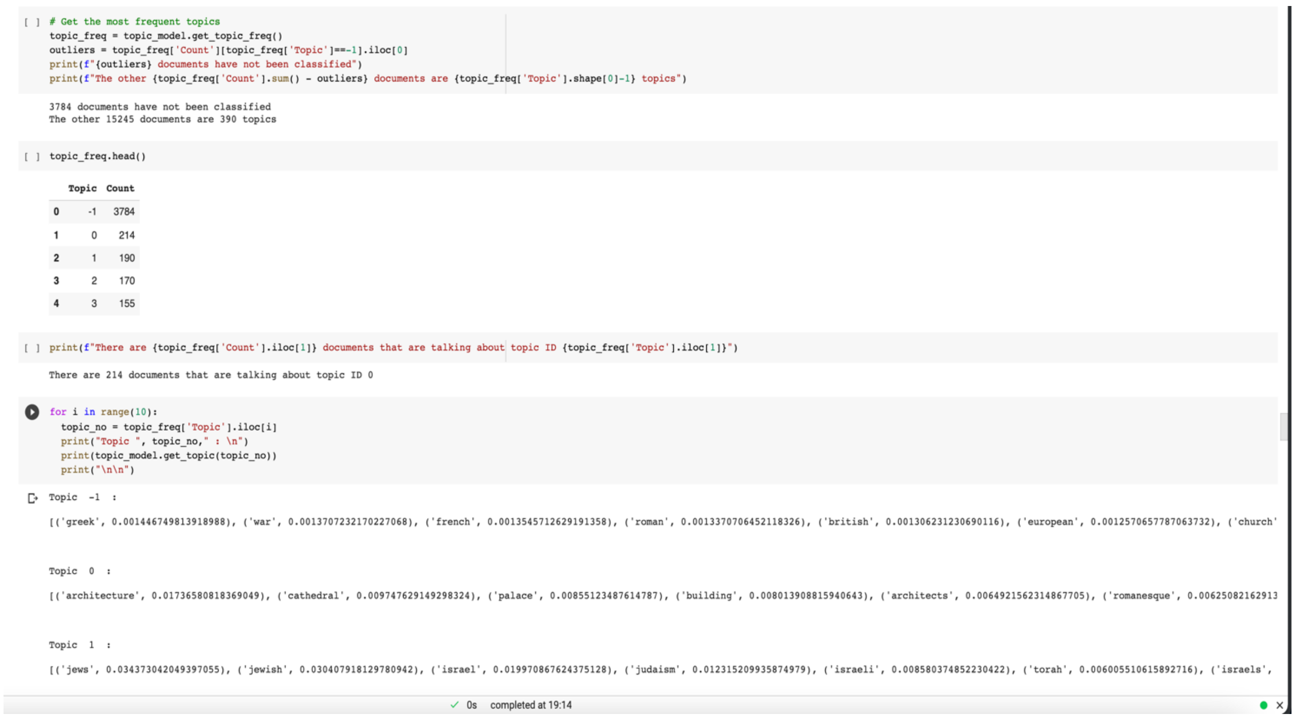
Date: 19 / 07 / 2021
Dataset Analyzed: SQUAD V2.0
Topic Model before removing stop words and lemmatization



```
[ ] # Get the most frequent topics
    topic_freq = topic_model.get_topic_freq()
    outliers = topic_freq['Count'][topic_freq['Topic']==-1].iloc[0]
    print(f"{outliers} documents have not been classified")
    print(f"The other {topic_freq['Count'].sum() - outliers} documents are {topic_freq['Topic'].shape[0]-1} topics")

    3784 documents have not been classified
    The other 15245 documents are 390 topics
```

```
[ ] topic_freq.head()
```

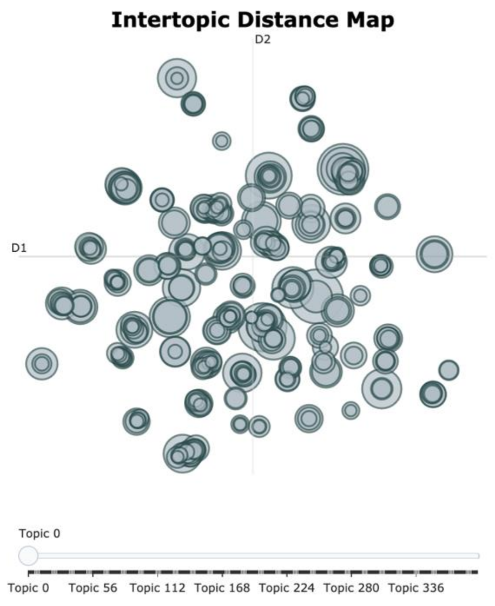
	Topic	Count
0	-1	3784
1	0	214
2	1	190
3	2	170
4	3	155

```
[ ] print(f"There are {topic_freq['Count'].iloc[1]} documents that are talking about topic ID {topic_freq['Topic'].iloc[1]}")

    There are 214 documents that are talking about topic ID 0
```

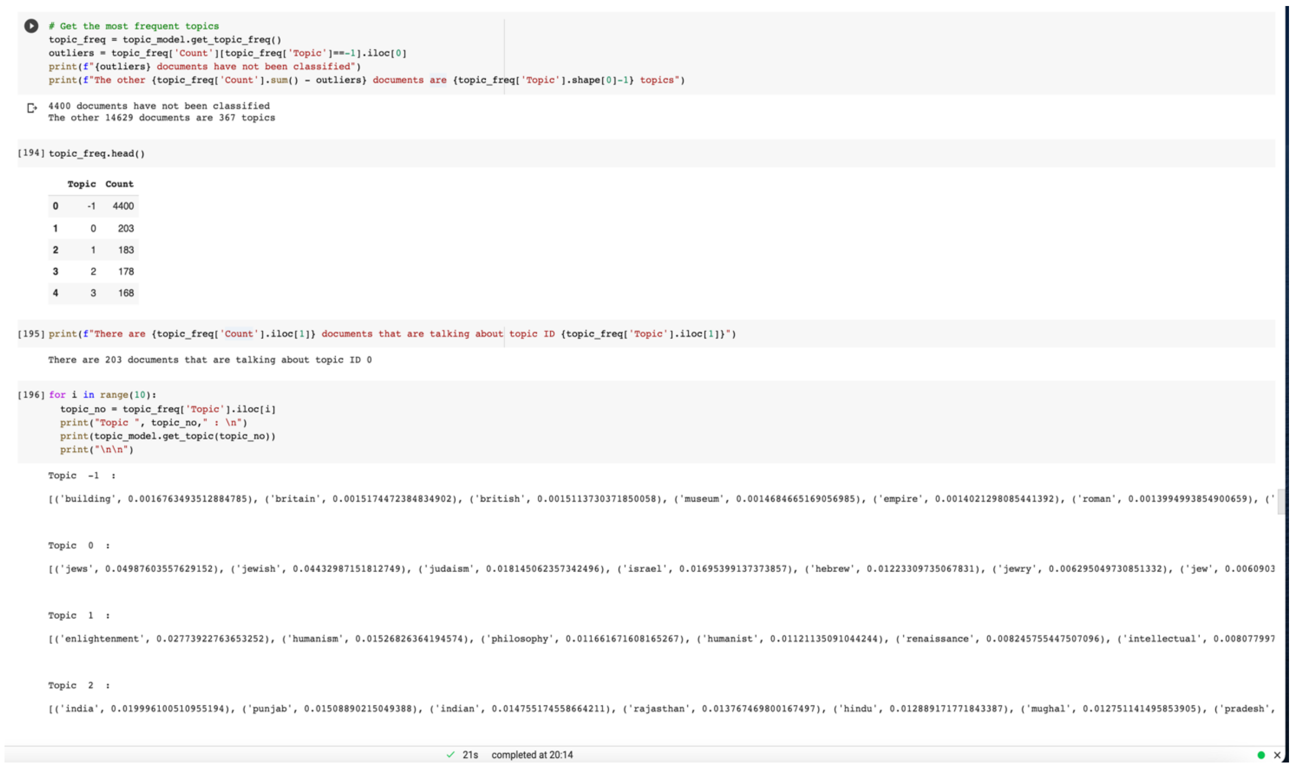
```
    for i in range(10):
      topic_no = topic_freq['Topic'].iloc[i]
      print("Topic ", topic_no," : \n")
      print(topic_model.get_topic(topic_no))
      print("\n\n")

    Topic  -1  :

    [('greek', 0.001446749813918988), ('war', 0.0013707232170227068), ('french', 0.0013545712629191358), ('roman', 0.0013370706452118326), ('british', 0.001306231230690116), ('european', 0.0012570657787063732), ('church'


    Topic  0  :

    [('architecture', 0.01736580818369049), ('cathedral', 0.009747629149298324), ('palace', 0.00855123487614787), ('building', 0.008013908815940643), ('architects', 0.0064921562314867705), ('romanesque', 0.00625082162913


    Topic  1  :

    [('jews', 0.034373042049397055), ('jewish', 0.030407918129780942), ('israel', 0.019970867624375128), ('judaism', 0.012315209935874979), ('israeli', 0.008580374852230422), ('torah', 0.006005510615892716), ('israels',
```

✓ 0s completed at 19:14

```
    # Returns how spatially separated each topic is.
    topic_model.visualize_topics()
```



Topic Model after removing stop words and lemmatization



```
# Get the most frequent topics
topic_freq = topic_model.get_topic_freq()
outliers = topic_freq['Count'][topic_freq['Topic']==-1].iloc[0]
print(f"{outliers} documents have not been classified")
print(f"The other {topic_freq['Count'].sum() - outliers} documents are {topic_freq['Topic'].shape[0]-1} topics")
```

```
4400 documents have not been classified
The other 14629 documents are 367 topics
```

```
[194] topic_freq.head()
```

	Topic	Count
0	-1	4400
1	0	203
2	1	183
3	2	178
4	3	168

```
[195] print(f"There are {topic_freq['Count'].iloc[1]} documents that are talking about topic ID {topic_freq['Topic'].iloc[1]}")
```

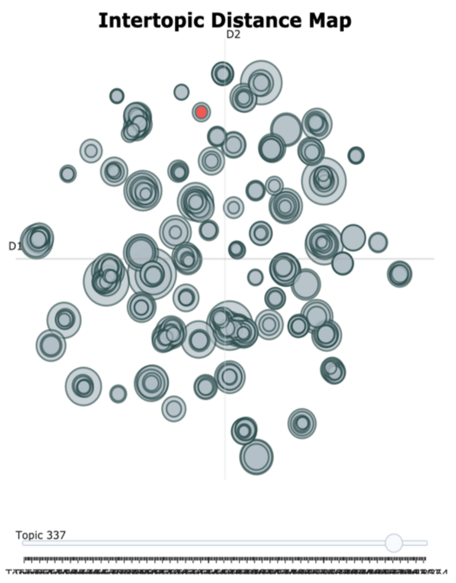
```
There are 203 documents that are talking about topic ID 0
```

```
[196] for i in range(10):
        topic_no = topic_freq['Topic'].iloc[i]
        print("Topic ", topic_no," : \n")
        print(topic_model.get_topic(topic_no))
        print("\n\n")
```

```
Topic  -1  :

[('building', 0.0016763493512884785), ('britain', 0.0015174472384834902), ('british', 0.0015113730371850058), ('museum', 0.0014684665169056985), ('empire', 0.0014021298085441392), ('roman', 0.0013994993854900659), ('


Topic  0  :

[('jews', 0.04987603557629152), ('jewish', 0.04432987151812749), ('judaism', 0.018145062357342496), ('israel', 0.01695399137373857), ('hebrew', 0.01223309735067831), ('jewry', 0.006295049730851332), ('jew', 0.0060903


Topic  1  :

[('enlightenment', 0.02773922763653252), ('humanism', 0.01526826364194574), ('philosophy', 0.011661671608165267), ('humanist', 0.01121135091044244), ('renaissance', 0.008245755447507096), ('intellectual', 0.008077997


Topic  2  :

[('india', 0.019996100510955194), ('punjab', 0.01508890215049388), ('indian', 0.014755174558664211), ('rajasthan', 0.013767469800167497), ('hindu', 0.012889171771843387), ('mughal', 0.012751141495853905), ('pradesh',
```

✓ 21s completed at 20:14

```
[197] # Returns how spatially separated each topic is.
      topic_model.visualize_topics()
```



Inferences:

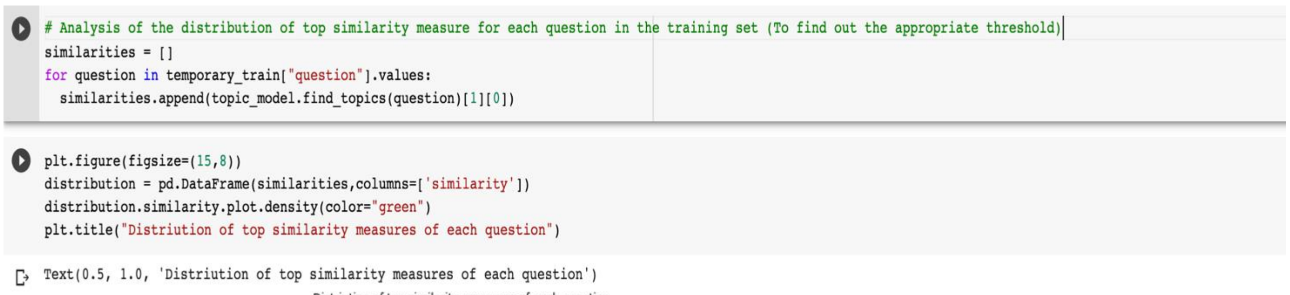
- Number of topics reduced from 390 to 367 → Some redundant clusters might have been removed due to preprocessing of contexts
- Number of documents that were not clustered have increased from 3784 to 4400.
- Intertopic Modelling Graph looks more spatially separated and still a lot of similar topics are clustered near the same region

Note: Spell check and correction wasn't performed as all available libraries (nltk,spacy,etc) perform a contextual spell check and the corrected word depends on the position it is present in the sentence (which supposedly seems like a good thing). But on actually performing it, it makes the dataset worse by changing words in a weird manner. Eg: "hoe" is changed to "why" and "which" is changed to "rice". It also removes many important words from the context completely.

Since the proportion of misspelt words are extremely less compared to the size of the dataset, we remove this step. (Else the correct contexts in the original dataset would get hampered and it would subsequently affect our overall performance)

Distribution of similarity measure (top 1 out of the list of 5) for each question returned by the Topic Model

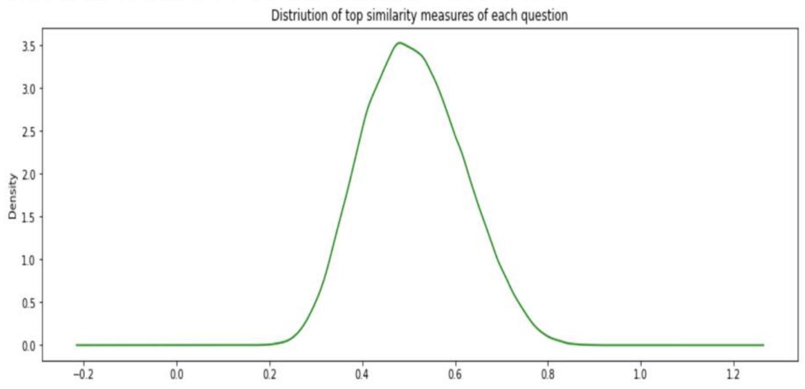
Density Plot of Similarities



```
# Analysis of the distribution of top similarity measure for each question in the training set (To find out the appropriate threshold)
similarities = []
for question in temporary_train["question"].values:
  similarities.append(topic_model.find_topics(question)[1][0])
```

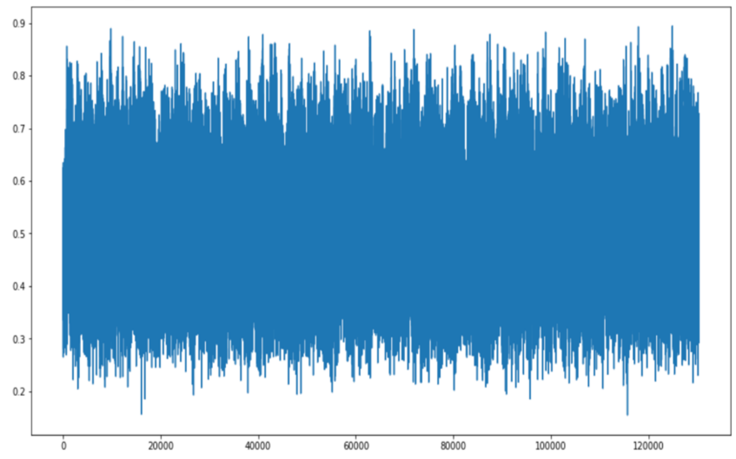
```
plt.figure(figsize=(15,8))
distribution = pd.DataFrame(similarities,columns=['similarity'])
distribution.similarity.plot.density(color="green")
plt.title("Distriution of top similarity measures of each question")
```

```
Text(0.5, 1.0, 'Distriution of top similarity measures of each question')
```



Overall Distribution of Similarities

```
[225] plt.figure(figsize=(15,8))
      distribution.similarity.plot()

      <matplotlib.axes._subplots.AxesSubplot at 0x7f289e1a7c90>
```



```
# Maximim similarity value for a question
print(max(similarities))
# Least similarity value for a question
print(min(similarities))

0.8944686866507052
0.1545840455889697
```

Mean Similarity value: 0.5105950028018558

Threshold Similarity Value	**Number of questions dropped in the original dataset**
0.1	**0**
0.2	**12**
0.3	**1868**
0.35	**7876**
0.4	**20584**
0.5	**62408**
0.6	**102930**

Inferences: Looking at the above number of records dropped and from analyzing the validity of the dropped questions, choosing a 0.3 - 0.4 value for threshold seems to be an ideal choice.

Note: Even after choosing a threshold value in this range, there are a good amount of dropped questions that still make a lot of sense and could be very well answered from the context paragraph.