

CS156 Final Project

[Introduction](#)

[Data Processing](#)

[K-means clustering](#)

[5 clusters](#)

[10 clusters](#)

[Selecting the best number of clusters](#)

[Visualizing K-means result](#)

[Topic modelling](#)

[Bags of word technique](#)

[TF-IDF technique](#)

[5 clusters](#)

[Finding the best number of clusters](#)

[Results](#)

[Conclusion](#)

[References](#)

[Appendices](#)

[HCS Appendix](#)

[Code](#)

Introduction

The dataset that we explored consist of a list of poll answers, their associated HCs/ LOs, and the score, for all the classes in Minerva in the past year. We used this dataset to explore two models: k-means clustering, and topic modeling using Latent Dirichlet Allocation (LDA) with either bags of words or term frequency (TF-IDF) technique, to explore potential hidden semantic spaces in the dataset that we explored.

Data Processing

The data is first downloaded and explored for assessment scores, poll responses, their means and counts. We discard the poll answers that are not scored, and when there are no LOs/ HCs associated with the poll responses. We then extract the text from the poll responses and tokenize and stem the words. Essentially, we split the response text into individual words, then convert them to their root form, (e.g. 'discuss', 'discussing', 'discussed' and 'discussion' all stem from the same word

discuss). We also filter the unnecessary words that does not matter in a clustering context (e.g. “I”, “we”, “a”, “the”).

Additionally, the dataset doesn't provide the associated colleges and classes for the poll response, so we need to map the poll response to the colleges ourselves by matching the LOs and HCs to the associated colleges. To do this, we make use of two additional dataframes provided in our Capstone Seminar classes - the LOs Master List and the HCs spreadsheet. We filter the name and the name, perform basic cleaning, then merge the HCs and LOs dataset together, following by an inner merge with the original dataset. Additionally, the mapping is incomplete because some of the HCs/ LOs change name. For instance, `#objectivemorality` becomes `#objmorality` while `#multiplecauses` become `#complexcausation`. To minimize the changes, we use fuzzy string mapping to find the best match for all of the unmatched HCs/ LOs (provided the match is at least 70%).

K-means clustering

We first do k-means clustering, which assigns each poll to a cluster based on their similarities, which is the Euclidean distance measured in the feature space. Before running the model, we transform the text to a TF-IDF matrix. TF-IDF (term frequency - inverse document frequency) is a numerical statistic that reflects how important a word is to a poll in all answers we get. The value increases as the number of time the word appears in the document but is offset by the number of time it appears in all documents, thus adjusting for the fact that some words just appear more frequently in general. This gives more accurate measurement for K-means clustering because it's not just dependent on words frequency but word frequency specific to the document itself. This can be seen from the word cloud visualization between bags of words and TF-IDF technique.

We see that both have similar common words (because, also, etc.), however TF-IDF information is more useful overall. Bags of words only show the most common words e.g. think, also, like, one, example, etc. which is used across major, whereas TF-IDF highlights things such as active, company, bias, argument, etc. which seem to be more specific to majors.

5 clusters

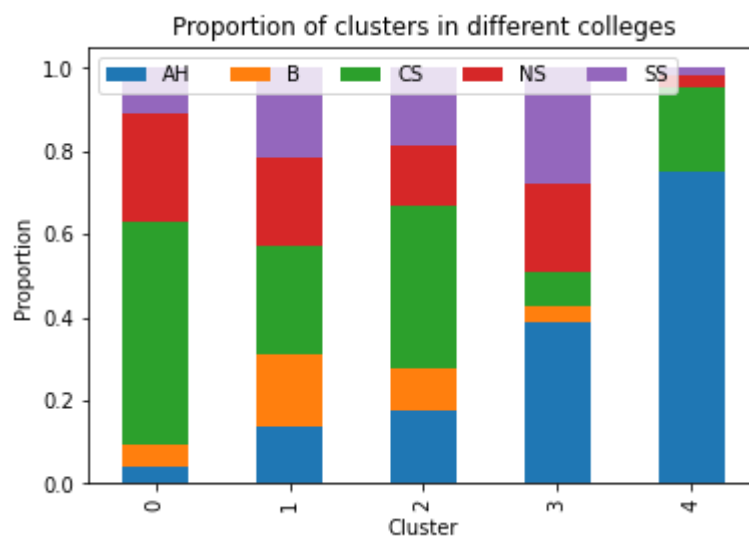
We first run a 5-cluster model, which is most appropriate because there are 5 colleges in Minerva, so the clusters would probably center around those words. We also get the 15 most common words from each clusters.

Top terms per cluster:

Cluster 0 words: data, variabl, model, sampl, use, distribut, would, test, hypothesi, probabl, observ, valu, mean, studi, differ,
Cluster 1 words: would, becaus, compani, market, use, time, one, valu, chang, product, increas, countri, also, need, could,
Cluster 2 words: poll, complet, student, present, fac, facil, facia, facial, facialex, facialexpress, facien, faciilit, facilitat, facevook, facili,
Cluster 3 words: use, think, differ, one, peopl, problem, would, understand, becaus, m ake, exampl, level, way, help, system,
Cluster 4 words: argument, thesi, sentenc, evid, induct, logic, deduct, premis, conclu s, true, statement, use, truth, claim, clone,

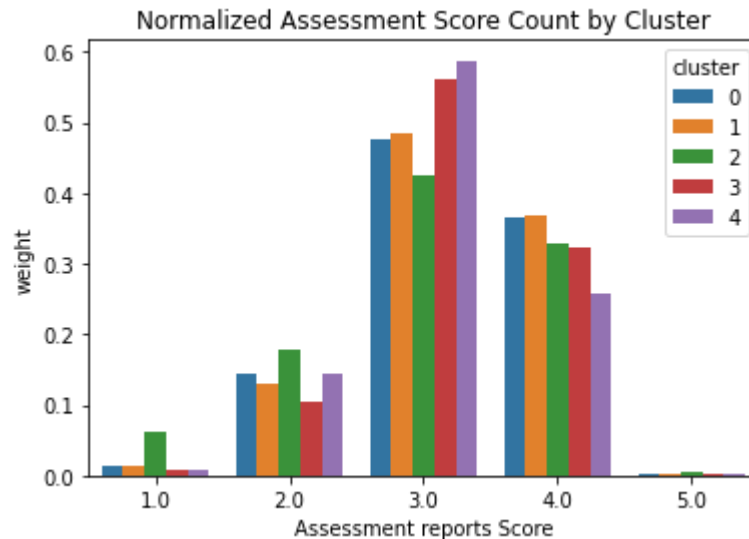
Based on the top term, it seems that cluster 0 maps best to CS major (with word like data, variable, distribution, etc.), cluster 1 maps best Business major (Company, market, product, country, etc.), cluster 3 seems to be the Social science cluster (people, system) and cluster 4 seems to be related to Computer Science formal analysis or Arts and Humanities major (premise, conclusion, logic, etc.).

Because we can map each poll response to a cluster, we can then map the proportion of colleges in each cluster.



We see that the mapping is pretty accurate for cluster 0 (majority CS) and cluster 4 (majority AH), but a bit more mixed for the other clusters. Note that Business is a small portion for all clusters because we don't have cornerstone course in Business, thus all the HCs scores will not have the Business college tag. As a result, the overall counts in Business is smaller.

We also use that to check the assessment score count by cluster, to see whether different clusters leads to different grade distribution.



We can see that clusters 0 and 1 is associated with higher ratio of 4s to 3s, while cluster 4 has the highest proportion for 3. It seems like Arts and Humanities professors are more likely to grade students 3s than CS professors!

There's also an interesting peak for score 1 for cluster 2, which is probably because the phrase "The student is present but did not complete the poll" is in this cluster (and are the most frequent words), which probably is associated with more 1.

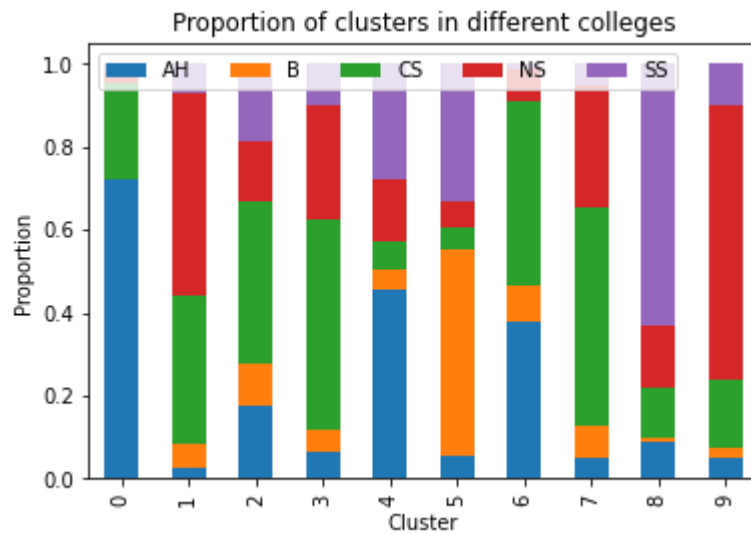
10 clusters

We also tried running the algorithm with 10 clusters instead of 5 and got the following results:

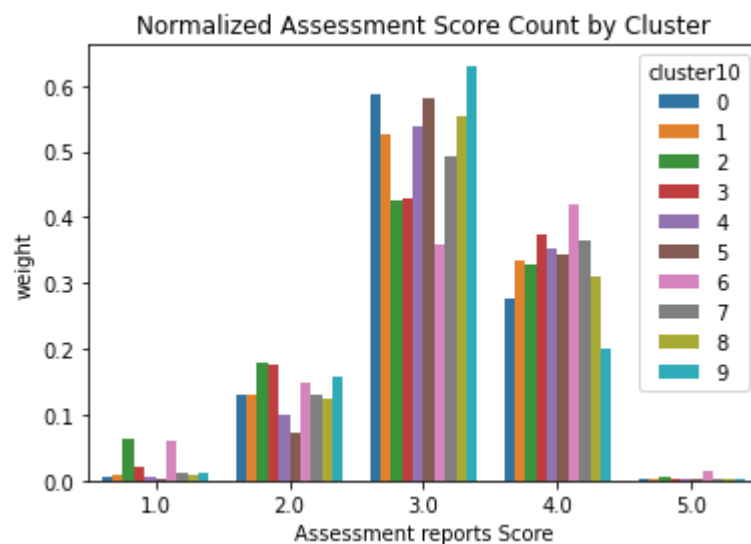
Top terms per cluster:

Cluster 0 words: argument, thesi, evid, induct, deduct, premis, conclus, use, statemen
t, claim, support, clone, reason, make, valid,
Cluster 1 words: variabl, studi, hypothesi, test, treatment, control, observ, group, e
ffect, experi, would, differ, confound, use, result,
Cluster 2 words: poll, complet, student, present, fac, facil, facia, facial, facialex,
facialexpress, facien, faciilit, facilit, facevook, facili,
Cluster 3 words: would, valu, use, probabl, number, time, becaus, function, distribut,
mean, one, model, sampl, first, get,
Cluster 4 words: peopl, use, think, would, one, becaus, differ, make, like, way, also,
examl, understand, could, work,
Cluster 5 words: compani, market, product, custom, risk, countri, busi, invest, cost,
economi, price, would, growth, increas, financi,
Cluster 6 words: doc, https, com, googl, edit, document, usp, share, kgi, edu, minerv
a, spreadsheet, drive, colab, gid,
Cluster 7 words: data, model, use, line, would, observ, graph, hypothesi, set, predic
t, collect, differ, regress, variabl, point,
Cluster 8 words: system, level, emerg, agent, interact, individu, properti, network, c
omplex, differ, predict, behavior, analysi, model, social,

Cluster 9 words: problem, solut, solv, constraint, water, use, rightproblem, identifi, state, differ, goal, breakdown, subproblem, step, one,



Interestingly, while the clusters contain more noise overall, some clusters do get better at classifying different colleges. We can more clearly identify cluster 0 as AH, cluster 9 as NS, cluster 5 as Business, cluster 8 as SS. We also get cluster 1 and 7 as a combination of NS and CS majors.

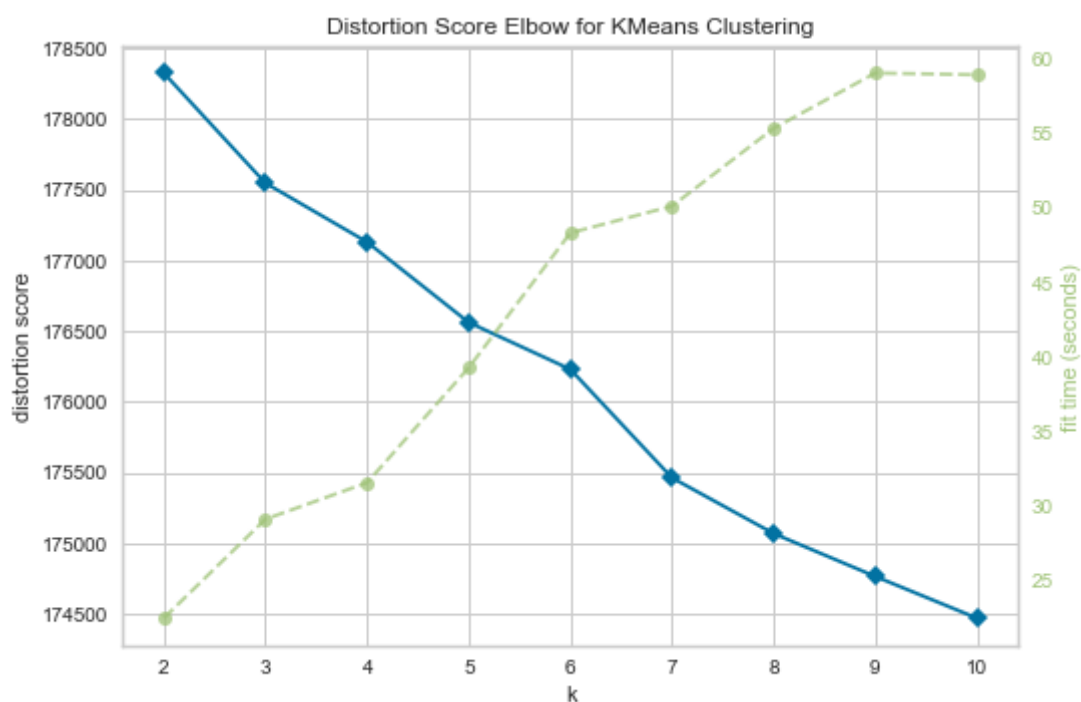


Once again, the normalized assessment count tells the same story: cluster 9 has the most 3s in proportion, followed by cluster 0, then 5, corresponding to NS, AH and Business major accordingly. Meanwhile, if the poll response is in cluster 6, it's more likely to get 4 and 5, but also more likely to get 1! The explanation is probably that the associated strings `doc, https, com, googl, edit, document` is probably associated

with when students have to paste the link of their pre-class work, which gives more time (and words) to get high scores, but can also get low scores if students are not well prepared.

Selecting the best number of clusters

We also run a K-Elbow Visualizer to find the best number of clusters. It runs the algorithm for all values of k for 2 to 10, then compute an average distortion score for all clusters, which is the sum of square distances from each point to its assigned center. We are looking for an inflexion point beyond which the distortion score stays constant.



While the elbow point is not clear, it seems that 7 is the best number of clusters. We run K-means again with 7 clusters to see if we get better results.

Top terms per cluster:

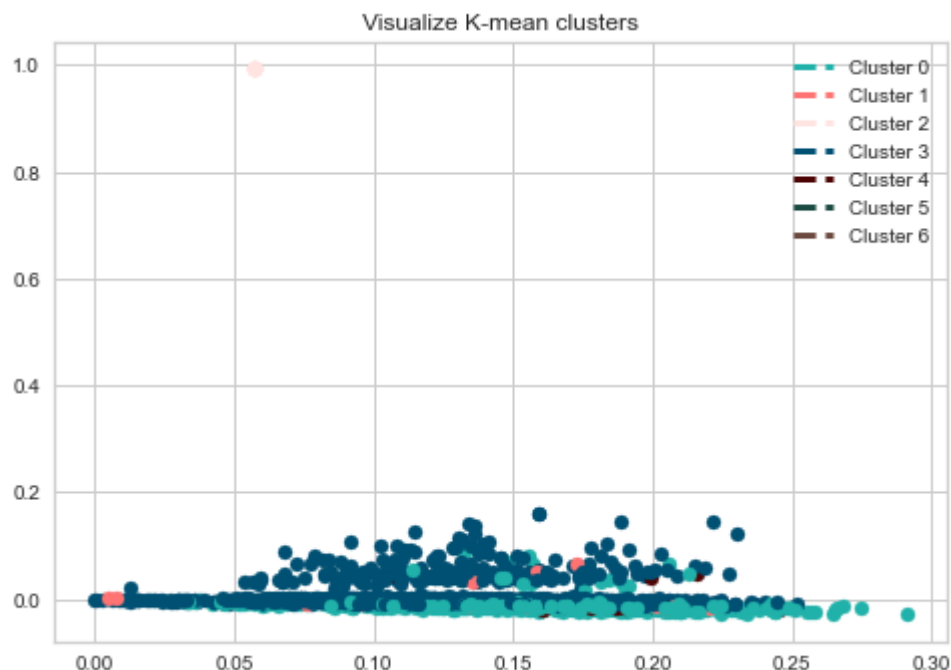
Cluster 0 words: problem, solut, solv, constraint, water, use, rightproblem, differ, i
dentifi, state, goal, breakdown, subproblem, step, one,
Cluster 1 words: poll, complet, student, present, faazillexzvnbceqmeprnivrzaadmewqcn
u, fabian, fabianokafor, fabiola, fabl, fabric, fabul, fac, faabi, facad, facbook,
Cluster 2 words: peopl, use, one, think, would, becaus, differ, make, exampl, argumen
t, understand, system, also, way, like,
Cluster 3 words: data, variabl, model, hypothesi, studi, observ, use, test, would, pre
dict, differ, control, treatment, one, regress,
Cluster 4 words: doc, https, com, googl, edit, document, usp, share, kgi, edu, minerv
a, spreadsheet, drive, colab, gid,

```
Cluster 5 words: compani, market, product, custom, risk, countri, busi, economi, cost,
invest, would, price, growth, financi, increas,
Cluster 6 words: would, valu, use, probabl, time, becaus, number, function, mean, dist
ribut, one, sampl, get, first, node,
```

There seems to be no significant improvement for 7 clusters compared to either the 5 or 10 clusters. It is also possible that the word clusters are not just dependent on majors, but also cultures, grades, English capacity, etc. But if we are judging based on the proportion of clusters in different colleges alone, the 7 clusters seem to do a mediocre job.

Visualizing K-means result

We wanted to do Principal Component Analysis and plot the clusters, but as the TF-IDF matrix is a sparse matrix, we used an alternative technique, `TruncatedSVD`, which performs dimensionality reduction for sparse matrices. Even after using this, however, our matrix was still too heavy, so we slice to 3000 datapoints. However, the result doesn't seem to separate as well as we have hoped for. This might be due to the slicing, or a non-optimal the dimension axes used.



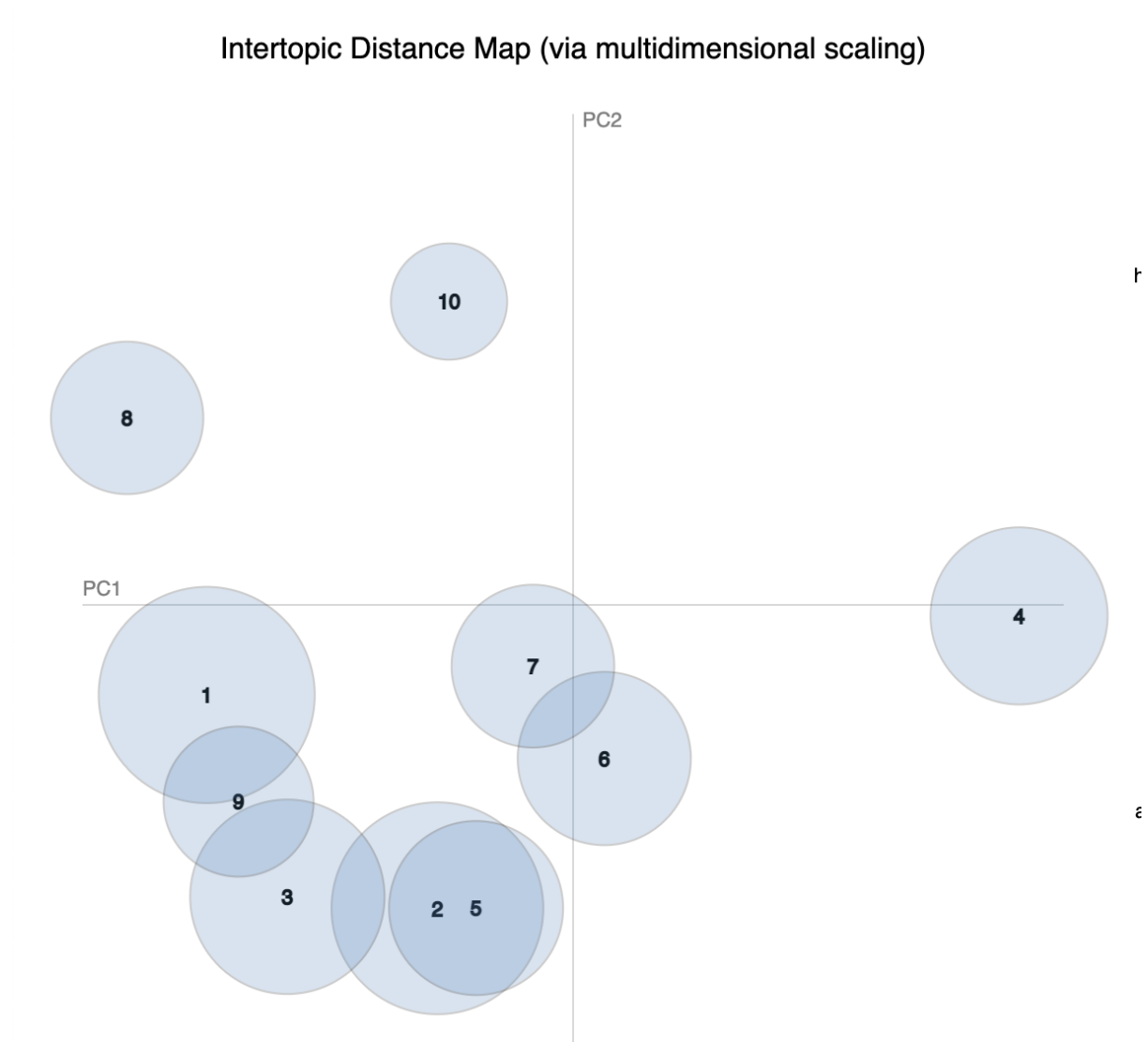
Topic modelling

K-means clustering problem is that it only assign each poll response to one cluster. Topic modeling allows for more flexibility by assigning each document to a probability

of belonging to each topic. We use Latent Dirichlet Allocation (LDA) with both bags of words or term frequency (TF-IDF) technique.

Bags of word technique

Bags of word is essentially using the cleaned response directly and mapping it to the LDA model. As a result, our clusters are not very separated.



We can also see the most frequent word within each cluster. This is determined by a parameter λ , which is between 0 and 1. The closer λ is to 1, the values will be ranked according to their frequency in the overall paper (without being exclusive to only the cluster itself), whereas the closer λ is to 0, the values will be ranked according to how exclusive they are to the cluster. That means the higher the λ , the more representative the sample (since it's determined by the frequency), but the less descriptive it is. Meanwhile, smaller λ can highlight interesting connections, but also

can be noisier. Thus, we choose λ to be equal to 0.5 to balance between these two considerations.

For instance, for cluster 4, the most common values seem to be variables in math equations, suggesting this is a cluster for math responses (as calculation, regression, matrix, equation, vector are all present). Cluster 8, meanwhile, have terms like emotion, music, poem, etc. which all tend to the Arts and Humanities college. Other clusters, however, are not as clear. For instance, cluster 1 has words like use, make, context, argument, moral, truth, which suggests that it might be Arts and Humanities, but the words are too general to be useful overall. Similarly, cluster 9 has social and group, culture, which makes us think it might be AH, but also has contrarian (a social science HC) and narwhal (a CS reference).

TF-IDF technique

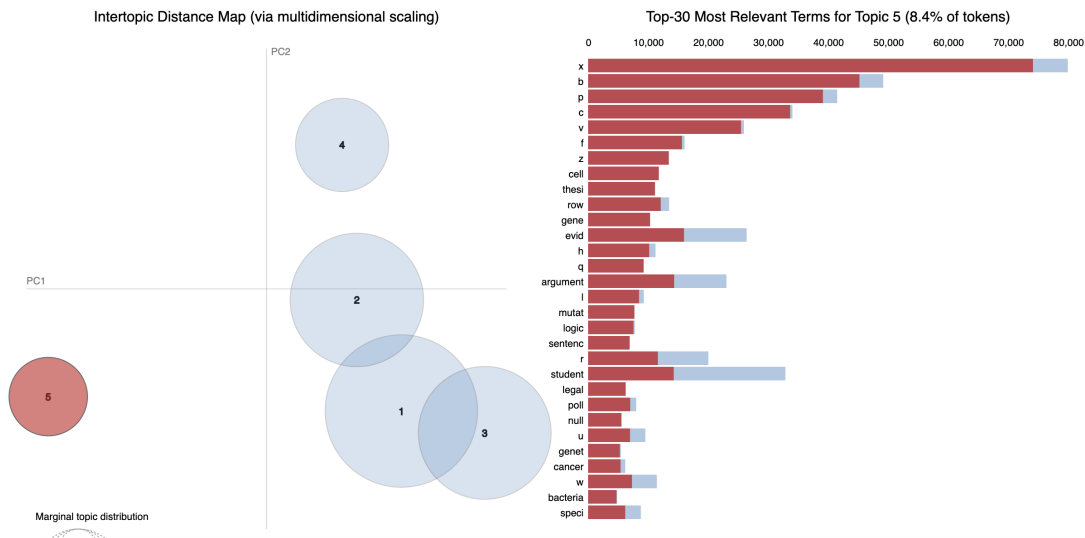
We try the topic modeling again with TF-IDF technique, which takes into account the frequency of the word in each response offset by overall frequency in all the polls. We were able to achieve much better separation, partly because the technique itself makes for a better measurement.

Intertopic Distance Map (via multidimensional scaling)



Here, we see a few clear clusters. For instance, 1 contains 'company', 'market', 'product', economy, business, growth, price, brand, all of which suggest it's a business-related cluster. 2 meanwhile has audience, music, story, narrative, thesis, art, etc. which is a humanity cluster. Cluster 6 contains p, n, treatment, hypothesis, sampling, which means this is probably either statistics or natural science. There's also gist and github which suggests it's more likely to be a statistics/ CS cluster. 9 has mineral, earth, atmosphere, ant, carbon, etc. suggesting it's a NS cluster, while 4 is likely to be a mathematics-related cluster (x, vector, equation, etc.). The other clusters are to be harder to predict, as they contain a mix bag of results.

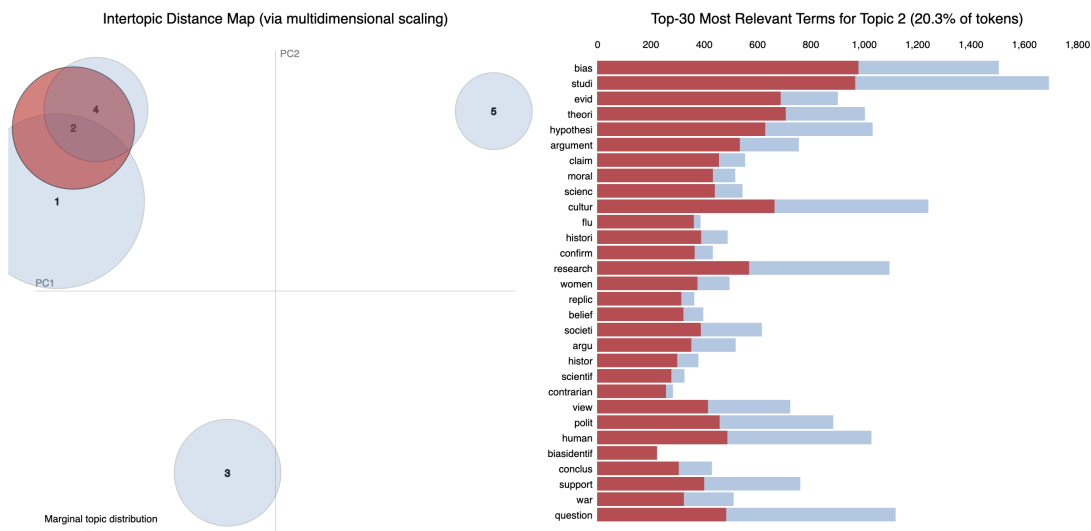
So maybe a problem might be that we have too many clusters, which make the results more diluted. We tried the topic modeling again with 5 clusters.



Once again, the result is not that clear with bag-of-words technique. The only differentiated cluster (5) is related to mathematics, but other clusters such as 4 and 2 also appear to be Computer science clusters. 3 is more likely to be business, but words like ‘policy’, ‘develop’, ‘resource’ and ‘economy’ suggests it can also be a mixture with the Economics social science classes.

5 clusters

The TF-IDF technique however, does a lot better with just 5 clusters.



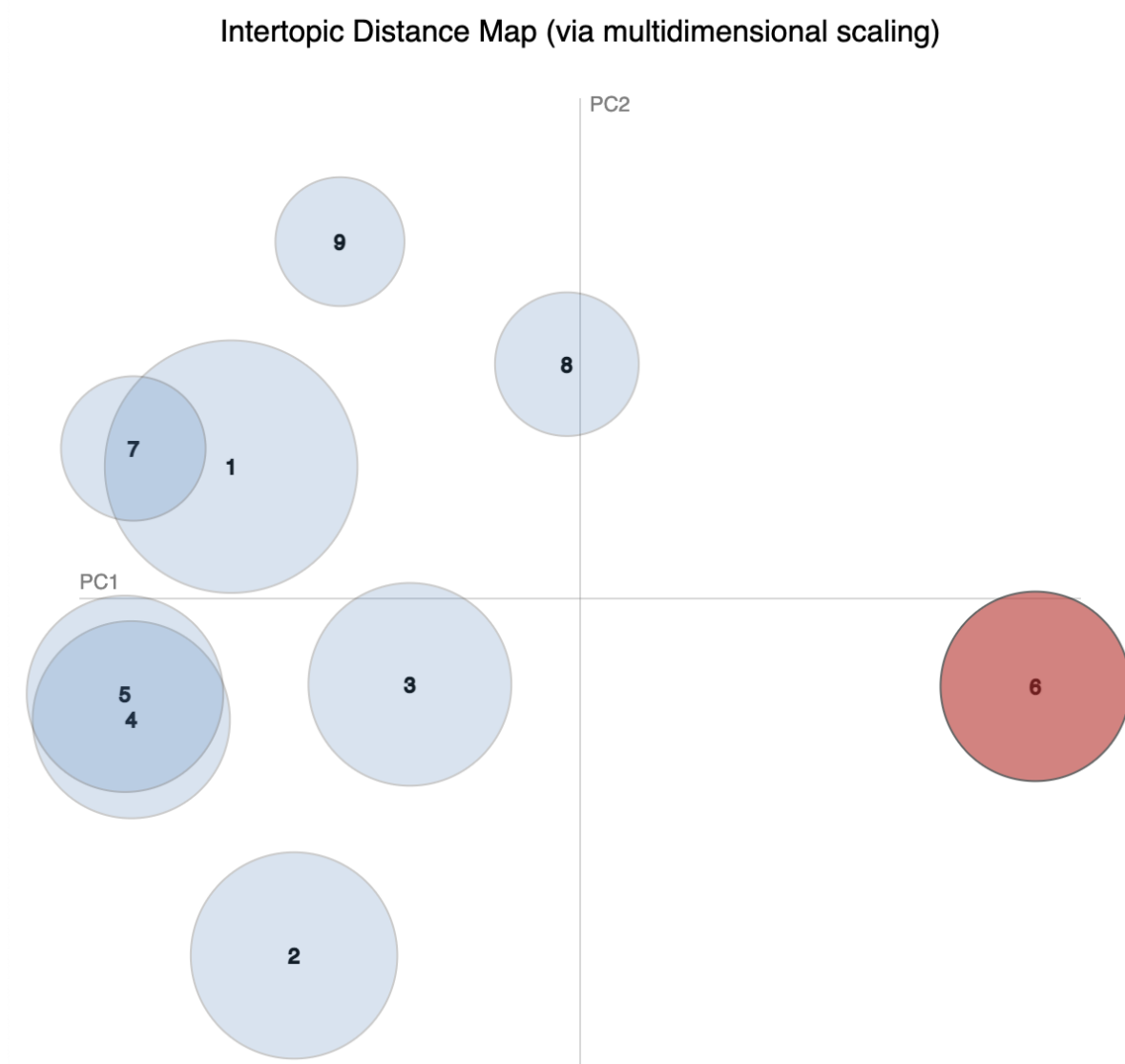
Even though the clusters do not look as separated, they actually map really well to the colleges. Cluster 5 contains “cells”, “gene”, “energy”, “entropy”, “mutation”, suggesting it’s a Natural science cluster, while cluster 3 contains the different variables for mathematics, as well as algorithm, google docs links, which is typical of a Computer science cluster. 1, 2 and 4 overlap a lot, but are actually pretty

distinctive. 1 includes “company”, “market”, “product”, “strategy”, which suggests it’s clearly a Business cluster, while 4 has “music”, “audience”, “language”, “emotion”, “words” which makes it a Arts and Humanities cluster. 2 is the most unclear, with word such as “bias”, “study”, “evidence” “theory” which can belong to any major, but also there’s words like “culture”, “history”, “women”, “politics”, “war” that makes it somewhat indicative of Social science. Also, since social science seems to be the most multi-disciplinary, it’s probably harder to classify.

Finding the best number of clusters

To select the best number of cluster, we use coherence score to measure how interpretable the topics are to humans by measuring how similar the words are to each other. We use the CV coherence score, which is the default metric for valuation.

We first compare the coherence score between bags of words and TF-IDF and get a slightly better coherence score for TF-IDF (0.3885 vs 0.3381), which aligns with our observations when examining the data manually. Next we test the coherence score with different number of clusters, from 2 to 10, and get the highest score for 9 clusters (0.395).



It's interesting that from PCA, it doesn't seem better separated. Looking at the clusters individually, we do see the differences e.g. 6 is a Math/ CS cluster, 2 is Business cluster, 3 is a SS complex system cluster, 1 seems to be a mixture between links and hypothesis testing/ scientific method, 5 is a politics SS cluster, and 4 is more of a economics SS cluster. Others are more difficult to differentiate (e.g 8 is both CS/ SS brain major, 7 seems to be AH but also have Github inside, and 9 just seems completely random), and there's no clear cluster for NS major. Once again, judging from observation, it seems that the 9 clusters do not do better than the 5 clusters.

Results

The results are shown in an interactive website [here](#).

Conclusion

K-means clustering and LDA provide two interesting ways of looking at the same set of data and finding clusters of words surrounding the topic of interest. It's hard to judge their accuracy beyond human examination, but it's interesting that both seem to pick up on the same patterns of words especially for CS, Business and AH majors, but struggle more with NS and SS.

The next finding we have is that finding optimal number of clusters are not as accurate as we have expected, especially when comparing with manually chosen clusters. It's probably because in our cases, we already have a predisposition as to how many clusters the words should have, which is more accurate than the calculation of an approximation of the metrics we want. It's also possible that because we don't have a clear metric of success beyond "looking at the words and seeing if the clusters are interpretable", thus manually chosen clusters might feel more accurate despite performing objectively better in certain metrics if we had calculated. But since natural language processing is just an extension of human language processing, and given we are working with manageable datasets, we feel this is sufficient for exploration, but can potentially made more rigorous if we want to do further hypothesis testing.

Another interesting finding is that, judging by observation alone, k-means seems to get more concentrated clusters with higher number of clusters, while LDA seems to get better results with fewer clusters. A potential explanation is that since k-means only assign each response to 1 cluster, the more clusters there are, the better divided the space is, and thus some clusters get more accurate (even though you also get more noises in other clusters). Meanwhile, LDA get better response when there are fewer clusters due to the percentage assigned model - the more clusters, the more diluted the percentage, the more inaccurate the result will be.

In conclusion, this is an interesting exploration of topic modeling. In the process, we have learnt that 'coffee' and 'starbuck' is highly associated with CS majors, and 'vietnamese' is a word that frequently appears in the AH cluster. The things you find when you look at words closely...

References

BoW Model and TF-IDF For Creating Feature From Text. (2020, February 27). *Analytics Vidhya*. <https://www.analyticsvidhya.com/blog/2020/02/quick-introduction-bag-of-words-bow-tf-idf/>

CS221. (n.d.). Retrieved April 21, 2022, from <https://stanford.edu/~cpiech/cs221/handouts/kmeans.html>

Kapadia, S. (2020, December 29). *Evaluate Topic Models: Latent Dirichlet Allocation (LDA)*. Medium. <https://towardsdatascience.com/evaluate-topic-model-in-python-latent-dirichlet-allocation-lda-7d57484bb5d0>

Kulshrestha, R. (2020, September 28). *Latent Dirichlet Allocation(LDA)*. Medium. <https://towardsdatascience.com/latent-dirichlet-allocation-lda-9d1cd064ffa2>

LDavis Demo. (n.d.). Retrieved April 21, 2022, from <http://www.kennyshirley.com/LDAvis/#topic=0&lambda=0.01&term=>

Zvornicanin, E. (2021, December 7). *When Coherence Score is Good or Bad in Topic Modeling? | Baeldung on Computer Science*. <https://www.baeldung.com/cs/topic-modeling-coherence-score>

Appendices

HCS Appendix

- *#differences*: One thing that makes the team efforts works really well is because we leverage our different abilities and skills to help each other complete the work. Esther is amazing at data science, telling stories and doing complex models, so she sets the direction for the project, thinks of different models to try and does a lot of the technical work. Ha is weaker in data science but stronger in writing and software engineering, so she helps with data cleaning and exploration, interprets the findings, writes the report and cleans up the website and Github. By focusing on each person's strength, we are able to create a coherent and clear report with a lot of technical analysis to back up.
- *#responsibility*: We were both very committed and proactive in the assignment. This is especially difficult since Esther is currently in Taiwan, and Ha is in San Francisco, so Esther goes to sleep when Ha starts working, and vice versa. To make this happen, we do check-in at the beginning and end of each person's work day, leave many messages and updates along the day. We also check-in with prof to make sure we are on the right track and get feedback for our final project.

Code

All accompanied code is found on this [Github repo](#). We have not added the dataset in the repo for security reason, but it will be in the zip files uploaded in the assignment submission.

We also downloaded the code and added it as PDF file in the next page.

data_processing

April 23, 2022

1 Data Processing

1.1 Data cleaning

```
[ ]: import pandas as pd
      from fuzzywuzzy import fuzz, process
      raw_df = pd.read_excel (r'/Users/swimmingcircle/cs156_code/assignments/
      ↪Final_project/long_form_poll_responses_2019.xlsx')
```

```
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/fuzzywuzzy/fuzz.py:11: UserWarning: Using slow pure-python
SequenceMatcher. Install python-Levenshtein to remove this warning
      warnings.warn('Using slow pure-python SequenceMatcher. Install python-
Levenshtein to remove this warning')
```

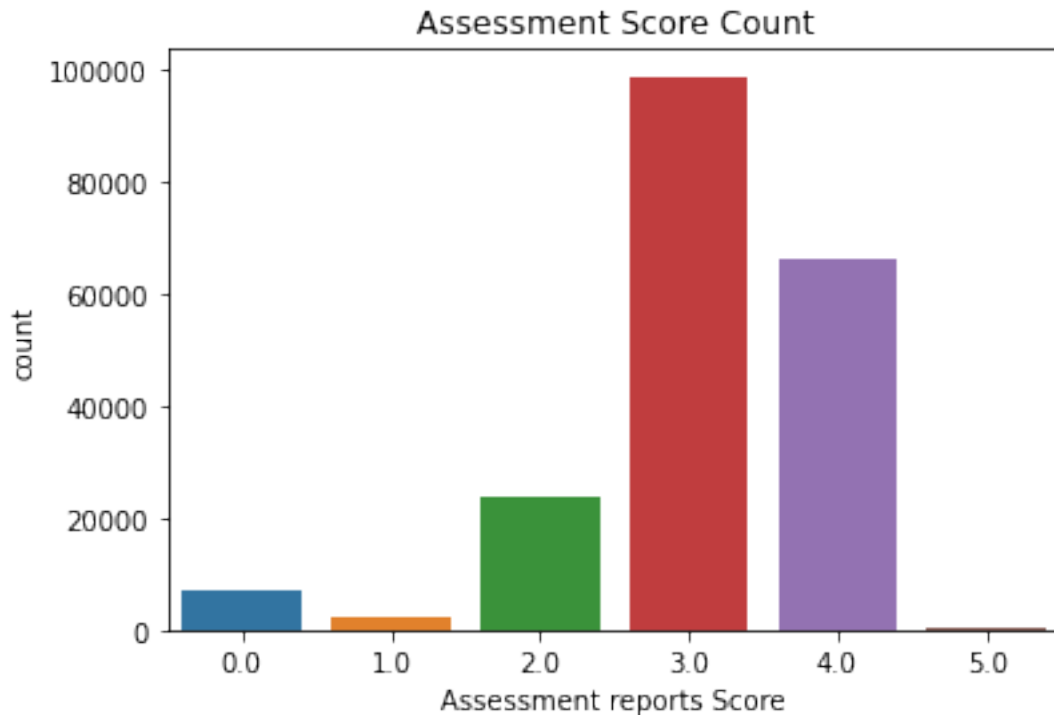
```
[ ]: #add time stamp as variable
      raw_df['time_stamp'] = raw_df['Polls ID'].rank(method = 'dense',
      ↪ascending=True).astype(int)
      raw_df.sort_values("time_stamp", inplace = True)
```

```
[ ]: import matplotlib.pyplot as plt
      import seaborn as sns
```

```
[ ]: sns.countplot(raw_df['Assessment reports Score'])
      plt.title('Assessment Score Count')
```

```
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable
as a keyword arg: x. From version 0.12, the only valid positional argument will
be `data`, and passing other arguments without an explicit keyword will result
in an error or misinterpretation.
      warnings.warn(
```

```
[ ]: Text(0.5, 1.0, 'Assessment Score Count')
```

```
[ ]: #average
raw_df['Assessment reports Score'].mean()
```

```
[ ]: 3.0913772213247173
```

We can see that most polls are graded as 3, and more polls are graded as 4 than 2. In addition, we see 0 in the grade, which we will need to discard. In addition, the average poll grades is around 3.09.

1.2 Grades for hashtags(HCs & LOs)

```
[ ]: avg_grades = pd.DataFrame(raw_df.groupby('Assessment reports_
↳Hashtag')['Assessment reports Score'].agg(['mean', 'count']))
avg_grades = avg_grades.sort_values('mean', ascending= False).reset_index()
```

```
[ ]: avg_grades
```

```
[ ]:
Assessment reports Hashtag    mean  count
0      #histonemodifications  4.000000     2
1      #LPimplementation     4.000000     1
2      #networktheory         4.000000     2
3      #Participation          4.000000     3
4      #Preparedness           4.000000     1
..      ...                   ...     ...
```

798	#buckinghampi	2.307692	39
799	#DA	2.285714	49
800	#Rimplementation	2.178571	28
801	#accountability	1.750000	8
802	NONE	0.040808	7082

[803 rows x 3 columns]

From the result, we can see that some hashtags have a high mean because of few data points. We might discard those hashtags if they bias the result. We will preserve the information for now.

1.3 Discard data

- Discard polls that have NONE for Assessment reports Hashtag
- Discard polls that have't get graded

```
[ ]: df = raw_df[raw_df['Assessment reports Hashtag'] != 'NONE']
df = raw_df[raw_df['Assessment reports Score'] != 0.0]
```

```
[ ]: df
```

```
[ ]:
      Polls ID  Assessment reports Student ID \
0         12522.0                41.0
16        12522.0               335.0
15        12522.0               333.0
14        12522.0               318.0
13        12522.0               315.0
...         ...                ...
198045    322265.0               1120.0
198046    322265.0               1121.0
198047    322265.0               1126.0
198034    322265.0                719.0
198079    322265.0               9357.0
```

	Poll Responses	Response \
0	The strengths of Plato's approach is his const...	
16	In the breakout we discussed if outside the ca...	
15	I think he's good at reasoning, but he has thi...	
14	Back to cmmom confusion time: the section 'und...	
13	Most difficult weakness is that his position w...	
...
198045		Definiteness
198046	I think important propoerties of algorithms in...	
198047	This is maybe not an additional property, but ...	
198034	comprehensibility. This is related to efficie...	
198079	Adaptability: I'm not sure if it's actually ap...	

Assessment reports Hashtag	Assessment reports Score	time_stamp
----------------------------	--------------------------	------------

0	#objectivemorality	2.0	1
16	#objectivemorality	3.0	1
15	#deductivearg	2.0	1
14	#objectivemorality	2.0	1
13	#objectivemorality	2.0	1
...
198045	#algorithmicstrategies	1.0	15539
198046	#algorithmicstrategies	3.0	15539
198047	#algorithmicstrategies	3.0	15539
198034	#algorithmicstrategies	3.0	15539
198079	#algorithmicstrategies	3.0	15539

[191016 rows x 6 columns]

1.4 Text processing

```
[ ]: import numpy as np
import pandas as pd
import nltk
from nltk.stem.snowball import SnowballStemmer
import re
import os
import codecs
from sklearn import feature_extraction
from nltk.tokenize import RegexpTokenizer
```

```
[ ]: # load nltk's English stopwords as variable called 'stopwords'
# use nltk.download() to install the corpus first
# Stop Words are words which do not contain important significance to be used
↳ in Search Queries
stopwords = nltk.corpus.stopwords.words('english')

# load nltk's SnowballStemmer as variable 'stemmer'
stemmer = SnowballStemmer("english")
```

```
[ ]: df['Poll Responses Response'] = df['Poll Responses Response'].apply(str)

#remove numbers
df['Poll Responses Response'] = df['Poll Responses Response'].str.
↳ replace(r'\d+', '')
```

/var/folders/0h/xyv81g2n7sj6zr0c9cw30gkc0000gn/T/ipykernel_17825/2124817980.py:1
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
df['Poll Responses Response'] = df['Poll Responses Response'].apply(str)
/var/folders/0h/xyv81g2n7sj6zr0c9cw30gkc0000gn/T/ipykernel_17825/2124817980.py:4
: FutureWarning: The default value of regex will change from True to False in a
future version.
```

```
df['Poll Responses Response'] = df['Poll Responses
Response'].str.replace(r'\d+', '')
/var/folders/0h/xyv81g2n7sj6zr0c9cw30gkc0000gn/T/ipykernel_17825/2124817980.py:4
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
df['Poll Responses Response'] = df['Poll Responses
Response'].str.replace(r'\d+', '')
```

2 Tokenize and Stem

We tokenize the stem the texts in the poll responses. - Tokenize: Split a sentence into words. We use `RegexTokenizer` to take out all the punctuations. - Stem: Reduce inflection in words to their root forms. For instance, playing, played, plays will all be presented as play. - Lemmatize: Consider the context and converts the word to its meaningful base form, which is called Lemma. The same word can have multiple different Lemmas. If you lemmatize the word ‘Stripes’ in verb context, it would return ‘Strip’. If you lemmatize it in noun context, it would return ‘Stripe’. If you just stem it, it would just return ‘Strip’.

We decide to use tokenize and stem as our first step to start. We then take out the stopwords, such as our, ours, same, so, than...

```
[ ]: tokenizer = RegexTokenizer("[\w']+")
df['tokenized_responses'] = df['Poll Responses Response'].map(tokenizer.
    ↪tokenize)
df['stemmed_responses'] = df['tokenized_responses'].apply(lambda x: [stemmer.
    ↪stem(y) for y in x])
```

```
/var/folders/0h/xyv81g2n7sj6zr0c9cw30gkc0000gn/T/ipykernel_17825/646613428.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
df['tokenized_responses'] = df['Poll Responses
Response'].map(tokenizer.tokenize)
/var/folders/0h/xyv81g2n7sj6zr0c9cw30gkc0000gn/T/ipykernel_17825/646613428.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
df['stemmed_responses'] = df['tokenized_responses'].apply(lambda x:
    [stemmer.stem(y) for y in x])
```

```
[ ]: df.head()
```

```
[ ]:      Polls ID  Assessment reports Student ID  \
0      12522.0                        41.0
16     12522.0                        335.0
15     12522.0                        333.0
14     12522.0                        318.0
13     12522.0                        315.0
```

```

                                Poll Responses Response  \
0  The strengths of Plato's approach is his const...
16 In the breakout we discussed if outside the ca...
15 I think he's good at reasoning, but he has thi...
14 Back to cmmon confusion time: the section 'und...
13 Most difficult weakness is that his position w...
```

```

Assessment reports Hashtag  Assessment reports Score  time_stamp  \
0      #objectivemorality                2.0           1
16     #objectivemorality                3.0           1
15      #deductivearg                2.0           1
14     #objectivemorality                2.0           1
13     #objectivemorality                2.0           1
```

```

                                tokenized_responses  \
0  [The, strengths, of, Plato's, approach, is, hi...
16 [In, the, breakout, we, discussed, if, outside...
15 [I, think, he's, good, at, reasoning, but, he,...
14 [Back, to, cmmon, confusion, time, the, sectio...
13 [Most, difficult, weakness, is, that, his, pos...
```

```

                                stemmed_responses
0  [the, strength, of, plato, approach, is, his, ...
16 [in, the, breakout, we, discuss, if, outsid, t...
15 [i, think, he, good, at, reason, but, he, has,...
14 [back, to, cmmon, confus, time, the, section, ...
13 [most, difficult, weak, is, that, his, posit, ...
```

```
[ ]: #take out stop words
df['clean_responses'] = df['stemmed_responses'].apply(lambda x: [item for item_
    ↪in x if item not in stopwords])
df['string'] = df['clean_responses'].apply(" ".join)
```

```
/var/folders/0h/xyv81g2n7sj6zr0c9cw30gkc0000gn/T/ipykernel_17825/3547584793.py:2
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
df['clean_responses'] = df['stemmed_responses'].apply(lambda x: [item for item
in x if item not in stopwords])
```

```
/var/folders/0h/xyv81g2n7sj6zr0c9cw30gkc0000gn/T/ipykernel_17825/3547584793.py:3
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
df['string'] = df['clean_responses'].apply(" ".join)
```

3 Add College & hashtag mapping data

We add two datasets, LOs and HCs sheets to map the HCs and LOs to their college in our dataframe df.

```
[ ]: #LO sheets
los = pd.read_csv("LOs Master List - LOs.csv")
print(los.shape)
los.head()
```

```
(622, 6)
```

```
[ ]: College Course LO (please exclude the hashtag) \
0 Arts & Humanities NaN NaN
1 AH AH110 globalruptures
2 AH AH110 globalpast
3 AH AH110 glocalization
4 AH AH110 historicaldebate
```

```

Rubric \
0 NaN
1 https://forum.minerva.edu/app/outcome-index/le...
2 https://forum.minerva.edu/app/outcome-index/le...
3 https://forum.minerva.edu/app/outcome-index/le...
4 https://forum.minerva.edu/app/outcome-index/le...
```

```

Description Last Update
0 NaN NaN
1 Analyze how economic, social, political, intel... Dec/2020
2 Place a historical phenomenon in global perspe... NaN
```

3	Analyze the mutually-impacting relationship be...	NaN
4	Analyze the way in which secondary sources on ...	NaN

```
[ ]: los = los[los['LO (please exclude the hashtag)'].notnull()]
los['LOs/ HCs'] = np.where(los['LO (please exclude the hashtag)'].str[0] == '#',
    los['LO (please exclude the hashtag)'].apply(str.lower).str[1:],
    los['LO (please exclude the hashtag)'].str.lower())
los = los.filter(['College', 'Course', 'LOs/ HCs'], axis = 1)

los.head()
```

```
[ ]: College Course      LOs/ HCs
1      AH AH110      globalruptures
2      AH AH110      globalpast
3      AH AH110      glocalization
4      AH AH110      historicaldebate
5      AH AH110      historicalperspective
```

```
[ ]: #HC sheet
hcs = pd.read_csv("HCs.csv")
hcs.head()
```

```
[ ]: HC (resources on the Hub) Course      Competency \
0      #confidenceintervals      FA Analyzing data
1      #correlation      FA Analyzing data
2      #descriptivestats      FA Analyzing data
3      #distributions      FA Analyzing data
4      #probability      FA Analyzing data

Brief description (own words)      Priority level for PRODUCT      Plans for use \
0      NaN      NaN      NaN
1      NaN      NaN      NaN
2      NaN      NaN      NaN
3      NaN      NaN      NaN
4      NaN      NaN      NaN

Updates on actual usage      Evidence of strong applications      Other notes
0      NaN      NaN      NaN
1      NaN      NaN      NaN
2      NaN      NaN      NaN
3      NaN      NaN      NaN
4      NaN      NaN      NaN
```

```
[ ]: hcs["College"] = hcs["Course"].map({"FA": "CS", "EA": "NS", "MC": "AH", "CX": "SS"})
hcs["LOs/ HCs"] = hcs["HC (resources on the Hub)"].apply(str.lower).str[2:]
```

```
hcs = hcs.filter(["LOs/ HCs", "College"])
hcs.head()
```

```
[ ]:      LOs/ HCs College
0  confidenceintervals    CS
1      correlation        CS
2  descriptivestats      CS
3  distributions        CS
4      probability      CS
```

```
[ ]: #concat LOs and HCs
hcs_and_los = pd.concat([hcs, los], axis = 0)
print(hcs_and_los.shape)
print(len(hcs_and_los["LOs/ HCs"].unique()))

#drop duplicates, because some classes have same LOs
hcs_and_los = hcs_and_los.drop_duplicates(subset='LOs/ HCs', keep='last')
print(hcs_and_los.shape)
```

```
(695, 3)
```

```
632
```

```
(632, 3)
```

```
[ ]: df['LOs/ HCs'] = np.where(df['Assessment reports Hashtag'].str[0] == '#',
    ↳df['Assessment reports Hashtag'].apply(str.lower).str[1:], df['Assessment_
    ↳reports Hashtag'].str.lower())
df = df[df["string"].notnull()]
df.head()
```

```
/var/folders/0h/xyv81g2n7sj6zr0c9cw30gkc0000gn/T/ipykernel_17825/4189087042.py:1
```

```
: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead
```

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
df['LOs/ HCs'] = np.where(df['Assessment reports Hashtag'].str[0] == '#',
df['Assessment reports Hashtag'].apply(str.lower).str[1:], df['Assessment
reports Hashtag'].str.lower())
```

```
[ ]:      Polls ID  Assessment reports Student ID  \
0      12522.0                41.0
16     12522.0                335.0
15     12522.0                333.0
14     12522.0                318.0
13     12522.0                315.0
```

```
Poll Responses Response  \
```



```

0 The strengths of Plato's approach is his const...
16 In the breakout we discussed if outside the ca...
15 I think he's good at reasoning, but he has thi...
14 Back to cmmon confusion time: the section 'und...
13 Most difficult weakness is that his position w...

```

	Assessment reports Hashtag	Assessment reports Score	time_stamp \
0	#objectivemorality	2.0	1
16	#objectivemorality	3.0	1
15	#deductivearg	2.0	1
14	#objectivemorality	2.0	1
13	#objectivemorality	2.0	1

	tokenized_responses \
0	[The, strengths, of, Plato's, approach, is, hi...
16	[In, the, breakout, we, discussed, if, outside...
15	[I, think, he's, good, at, reasoning, but, he,...
14	[Back, to, cmmon, confusion, time, the, sectio...
13	[Most, difficult, weakness, is, that, his, pos...

	stemmed_responses \
0	[the, strength, of, plato, approach, is, his, ...
16	[in, the, breakout, we, discuss, if, outsid, t...
15	[i, think, he, good, at, reason, but, he, has,...
14	[back, to, cmmon, confus, time, the, section, ...
13	[most, difficult, weak, is, that, his, posit, ...

	clean_responses \
0	[strength, plato, approach, construct, whole, ...
16	[breakout, discuss, outsid, cave, might, bigge...
15	[think, good, reason, circular, way, prove, po...
14	[back, cmmon, confus, time, section, understand...
13	[difficult, weak, posit, understand, testabl, ...

	string	L0s/ HCs
0	strength plato approach construct whole framew...	objectivemorality
16	breakout discuss outsid cave might bigger cave...	objectivemorality
15	think good reason circular way prove point onl...	deductivearg
14	back cmmon confus time section understand inte...	objectivemorality
13	difficult weak posit understand testabl like i...	objectivemorality

```

[ ]: outer_df = pd.merge(df, hcs_and_los, on='L0s/ HCs', how='outer')

print(outer_df.shape)
outer_df.head()

```

```

(191128, 13)

```

```
[ ]: Polls ID Assessment reports Student ID \
0 12522.0 41.0
1 12522.0 335.0
2 12522.0 318.0
3 12522.0 315.0
4 12522.0 297.0
```

```
Poll Responses Response \
0 The strengths of Plato's approach is his const...
1 In the breakout we discussed if outside the ca...
2 Back to cmmon confusion time: the section 'und...
3 Most difficult weakness is that his position w...
4 I'm still trying to understand the significanc...
```

```
Assessment reports Hashtag Assessment reports Score time_stamp \
0 #objectivemorality 2.0 1.0
1 #objectivemorality 3.0 1.0
2 #objectivemorality 2.0 1.0
3 #objectivemorality 2.0 1.0
4 #objectivemorality 2.0 1.0
```

```
tokenized_responses \
0 [The, strengths, of, Plato's, approach, is, hi...
1 [In, the, breakout, we, discussed, if, outside...
2 [Back, to, cmmon, confusion, time, the, sectio...
3 [Most, difficult, weakness, is, that, his, pos...
4 [I'm, still, trying, to, understand, the, sign...
```

```
stemmed_responses \
0 [the, strength, of, plato, approach, is, his, ...
1 [in, the, breakout, we, discuss, if, outsid, t...
2 [back, to, cmmon, confus, time, the, section, ...
3 [most, difficult, weak, is, that, his, posit, ...
4 [i'm, still, tri, to, understand, the, signifi...
```

```
clean_responses \
0 [strength, plato, approach, construct, whole, ...
1 [breakout, discuss, outsid, cave, might, bigge...
2 [back, cmmon, confus, time, section, understand...
3 [difficult, weak, posit, understand, testabl, ...
4 [i'm, still, tri, understand, signific, cave, ...
```

```
string L0s/ HCs \
0 strength plato approach construct whole framew... objectivemorality
1 breakout discuss outsid cave might bigger cave... objectivemorality
2 back cmmon confus time section understand inte... objectivemorality
3 difficult weak posit understand testabl like i... objectivemorality
```

4 i'm still tri understand signific cave analog ... objectivemorality

	College	Course
0	NaN	NaN
1	NaN	NaN
2	NaN	NaN
3	NaN	NaN
4	NaN	NaN

```
[ ]: #in original dataset
unmatched = outer_df[outer_df["College"].isna()]["LOs/ HCs"].unique()
print(sorted(unmatched)[:10])
print(len(unmatched))
```

```
[' theoreticallevelsofanalysis', 'accountability', 'accountingrules',
'agentsanddevelopment', 'aidtrade', 'alftools', 'algorithmicstrategies',
'analyticalenterprise', 'approxandscaling', 'artcommtheory']
279
```

```
[ ]: from fuzzywuzzy import process

for lo in unmatched:
    best_match = process.extractOne(lo, hcs_and_los["LOs/ HCs"])
    if best_match[1] <= 70:
        continue
    df.loc[df['LOs/ HCs'] == lo, 'LOs/ HCs'] = best_match[0]

df = df.merge(hcs_and_los, on='LOs/ HCs', how='inner')
df
```

	Polls ID	Assessment reports	Student ID \
0	12522.0		41.0
1	12522.0		335.0
2	12522.0		318.0
3	12522.0		315.0
4	12522.0		297.0
...
181936	320721.0		301.0
181937	320721.0		30.0
181938	320721.0		236.0
181939	320721.0		247.0
181940	321089.0		30.0

	Poll Responses	Response \
0	The strengths of Plato's approach is his const...	
1	In the breakout we discussed if outside the ca...	
2	Back to cmmon confusion time: the section 'und...	
3	Most difficult weakness is that his position w...	

4 I'm still trying to understand the significanc...

...

181936 Emailed it to you earlier. The explanation in ...

181937 <https://docs.google.com/document/d/JUbBIZWdCAW...>

181938 <https://docs.google.com/document/d/LBhIrfvdqPf...>

181939 . The free energy is $F = -NkT \ln(\cosh(dj \beta))$...

181940 I had seen some of these concepts before, but ...

	Assessment reports Hashtag	Assessment reports Score	time_stamp \
0	#objectivemorality	2.0	1
1	#objectivemorality	3.0	1
2	#objectivemorality	2.0	1
3	#objectivemorality	2.0	1
4	#objectivemorality	2.0	1
...
181936	#IsingModel	4.0	15312
181937	#IsingModel	4.0	15312
181938	#IsingModel	4.0	15312
181939	#IsingModel	4.0	15312
181940	#IsingModel	4.0	15334

	tokenized_responses \
0	[The, strengths, of, Plato's, approach, is, hi...
1	[In, the, breakout, we, discussed, if, outside...
2	[Back, to, cmmon, confusion, time, the, sectio...
3	[Most, difficult, weakness, is, that, his, pos...
4	[I'm, still, trying, to, understand, the, sign...
...	...
181936	[Emailed, it, to, you, earlier, The, explanati...
181937	[https, docs, google, com, document, d, JUbBIZ...
181938	[https, docs, google, com, document, d, LBhIrf...
181939	[The, free, energy, is, F, NkT, ln, cosh, dj, ...
181940	[I, had, seen, some, of, these, concepts, befo...

	stemmed_responses \
0	[the, strength, of, plato, approach, is, his, ...
1	[in, the, breakout, we, discuss, if, outsid, t...
2	[back, to, cmmon, confus, time, the, section, ...
3	[most, difficult, weak, is, that, his, posit, ...
4	[i'm, still, tri, to, understand, the, signifi...
...	...
181936	[email, it, to, you, earlier, the, explan, in,...
181937	[https, doc, googl, com, document, d, jubbizwd...
181938	[https, doc, googl, com, document, d, lbhirfvd...
181939	[the, free, energi, is, f, nkt, ln, cosh, dj, ...
181940	[i, had, seen, some, of, these, concept, befor...

```

                                clean_responses \
0      [strength, plato, approach, construct, whole, ...
1      [breakout, discuss, outsid, cave, might, bigge...
2      [back, cmmon, confus, time, section, understand...
3      [difficult, weak, posit, understand, testabl, ...
4      [i'm, still, tri, understand, signific, cave, ...
...
181936 [email, earlier, explan, cosh, express, larger...
181937 [https, doc, googl, com, document, jubbizwdcaw...
181938 [https, doc, googl, com, document, lbhirfvdqpf...
181939 [free, energi, f, nkt, ln, cosh, dj, beta, h, ...
181940 [seen, concept, befor, complet, rigor, grasp, ...

```

```

                                string      LOs/ HCs \
0      strength plato approach construct whole framew... objmorality
1      breakout discuss outsid cave might bigger cave... objmorality
2      back cmmon confus time section understand inte... objmorality
3      difficult weak posit understand testabl like i... objmorality
4      i'm still tri understand signific cave analog ... objmorality
...
181936 email earlier explan cosh express larger one t... isingmodel
181937 https doc googl com document jubbizwdcawhf_tnv... isingmodel
181938 https doc googl com document lbhirfvdqpfcf khu... isingmodel
181939 free energi f nkt ln cosh dj beta h thus unmag... isingmodel
181940 seen concept befor complet rigor grasp though ... isingmodel

```

```

College Course
0      AH  AH111
1      AH  AH111
2      AH  AH111
3      AH  AH111
4      AH  AH111
...
181936 NS  NS162
181937 NS  NS162
181938 NS  NS162
181939 NS  NS162
181940 NS  NS162

```

[181941 rows x 13 columns]

```

[ ]: #Check the result
df[df["College"].isna()]

```

[]: Empty DataFrame
Columns: [Polls ID, Assessment reports Student ID, Poll Responses Response, Assessment reports Hashtag, Assessment reports Score, time_stamp,

```
tokenized_responses, stemmed_responses, clean_responses, string, LOs/ HCs,  
College, Course]  
Index: []
```

```
[ ]: #convert into a csv  
compression_opts = dict(method='zip',  
                        archive_name='out.csv')  
df.to_csv('processed_response.zip', index=False,  
          compression=compression_opts)
```

Viz_K-means

April 23, 2022

1 Data Exploration & K-means

```
[ ]: import numpy as np
import pandas as pd
import sklearn
import seaborn as sns
import matplotlib.pyplot as plt
import random

random.seed(10)
```

```
[ ]: df = pd.read_csv('processed_response.zip')
```

```
[ ]: df.head()
```

```
[ ]:      Polls ID  Assessment reports Student ID \
0    12522.0                41.0
1    12522.0               335.0
2    12522.0               318.0
3    12522.0               315.0
4    12522.0               297.0
```

```
      Poll Responses Response \
0  The strengths of Plato's approach is his const...
1  In the breakout we discussed if outside the ca...
2  Back to cmmon confusion time: the section 'und...
3  Most difficult weakness is that his position w...
4  I'm still trying to understand the significanc...
```

```
      Assessment reports Hashtag  Assessment reports Score  time_stamp \
0      #objectivemorality          2.0              1
1      #objectivemorality          3.0              1
2      #objectivemorality          2.0              1
3      #objectivemorality          2.0              1
4      #objectivemorality          2.0              1
```

```
      tokenized_responses \
```

```

0 ['The', 'strengths', 'of', "Plato's", 'approac...
1 ['In', 'the', 'breakout', 'we', 'discussed', '...'
2 ['Back', 'to', 'cmmon', 'confusion', 'time', '...'
3 ['Most', 'difficult', 'weakness', 'is', 'that'...'
4 ["I'm", 'still', 'trying', 'to', 'understand', '...'

```

```

                                stemmed_responses \
0 ['the', 'strength', 'of', 'plato', 'approach', '...'
1 ['in', 'the', 'breakout', 'we', 'discuss', 'if...'
2 ['back', 'to', 'cmmon', 'confus', 'time', 'the...'
3 ['most', 'difficult', 'weak', 'is', 'that', 'h...'
4 ["i'm", 'still', 'tri', 'to', 'understand', 't...'

```

```

                                clean_responses \
0 ['strength', 'plato', 'approach', 'construct', '...'
1 ['breakout', 'discuss', 'outsid', 'cave', 'mig...'
2 ['back', 'cmmon', 'confus', 'time', 'section', '...'
3 ['difficult', 'weak', 'posit', 'understand', '...'
4 ["i'm", 'still', 'tri', 'understand', 'signifi...'

```

```

                                string      LOs/ HCs College \
0 strength plato approach construct whole framew... objmorality      AH
1 breakout discuss outsid cave might bigger cave... objmorality      AH
2 back cmmon confus time section understand inte... objmorality      AH
3 difficult weak posit understand testabl like i... objmorality      AH
4 i'm still tri understand signific cave analog ... objmorality      AH

```

```

Course
0 AH111
1 AH111
2 AH111
3 AH111
4 AH111

```

```
[ ]: df['string'] = df['string'].values.astype('U')
```

2 Transform dataframe into tfidf

Key points - Tf-idf computes weights of the words (how relevant they are in the document) - The higher the TF-IDF score, the rare or unique the term is, and vice versa - TF and IDF are calculated in different ways (either by ranking frequency values, or by dividing the words frequency overall by the number of words given in the dictionary)

Formula

- Term Frequency (TF):
 - Calculate the term frequency
 - $tf(t,d) = \text{count of } t \text{ in } d / \text{number of words in } d$

- Inverse Document Frequency(IDF)
 - Weigh down the frequent terms may appear a lot of times but have little importance
 - $\text{idf}(t) = \log(N/(\text{df} + 1))$
- Calculation: TF is multiplied by IDF

```
[ ]: from sklearn.feature_extraction.text import TfidfVectorizer
```

```
tfidf_vector = TfidfVectorizer()
tfidf_matrix = tfidf_vector.fit_transform(df['string'])
tfidf_matrix
```

```
[ ]: <181941x79400 sparse matrix of type '<class 'numpy.float64'>'
      with 6018555 stored elements in Compressed Sparse Row format>
```

```
[ ]: #Explore td-idf for one doc: print words & tf-idf scores
feature_names = tfidf_vector.get_feature_names()
doc = 3 #random doc
feature_index = tfidf_matrix[doc,:].nonzero()[1] #Return the indices of the
↳ elements that are non-zero.
tfidf_scores = zip(feature_index, [tfidf_matrix[doc, x] for x in feature_index])
for w, s in [(feature_names[i], s) for (i, s) in tfidf_scores]:
    print(w, s)
```

```
get 0.1602432886945829
way 0.14154666427634024
ani 0.18754112365767017
whether 0.1909802791293325
decid 0.22724769256321592
lot 0.1967549100540339
quit 0.2530798913029075
struggl 0.2884601285317333
peopl 0.13840463662592684
incomprehens 0.451983870307689
like 0.13508086186616905
testabl 0.27084504953544314
posit 0.1880356167579713
difficult 0.23321560468563032
understand 0.15017855307918143
cave 0.3964584008260598
weak 0.24596725240200004
```

```
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/sklearn/utils/deprecation.py:87: FutureWarning: Function
get_feature_names is deprecated; get_feature_names is deprecated in 1.0 and will
be removed in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

We explore words and their tf-idf scores for a random poll response in our dataset.

3.2 Visualize tf-idf

We visualize the word cloud for tf-idf matrix as well, since it takes a long time to run, we slice the matrix that remains around 20% (15000/total # of cols) of the information.

```
[ ]: tfidf_matrix
```

```
[ ]: <181941x79400 sparse matrix of type '<class 'numpy.float64'>'
      with 6018555 stored elements in Compressed Sparse Row format>
```

```
[ ]: response = tfidf_matrix[:, :15000] #slice the matrix
df_tfidf_sklearn = pd.DataFrame(response.toarray(), columns= tfidf_vector.
    ↳get_feature_names()[:15000])
df_tfidf_sklearn.head()
```

```
[ ]:
0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
1  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
2  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
3  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
4  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
```

```

----- ... conjunst conjunt conjur conjvey conlad conland \
0          0.0 ...      0.0      0.0      0.0      0.0      0.0
1          0.0 ...      0.0      0.0      0.0      0.0      0.0
2          0.0 ...      0.0      0.0      0.0      0.0      0.0
3          0.0 ...      0.0      0.0      0.0      0.0      0.0
4          0.0 ...      0.0      0.0      0.0      0.0      0.0
```

```

conlcud conlcus conley conflct
0      0.0      0.0      0.0      0.0
1      0.0      0.0      0.0      0.0
2      0.0      0.0      0.0      0.0
3      0.0      0.0      0.0      0.0
4      0.0      0.0      0.0      0.0
```

[5 rows x 15000 columns]

```
[ ]: #some up all frequency
tf_idf_counter = df_tfidf_sklearn.T.sum(axis=1)
```

```
[ ]: wordcloud = WordCloud(width = 500, height = 400, background_color="white",
    ↳max_words=5000)
wordcloud.generate_from_frequencies(tf_idf_counter)
wordcloud.to_image()
```

```
[ ]:
```



```
clusters = km.labels_.tolist()

df['cluster'] = np.array(clusters)
terms = tfidf_vector.get_feature_names_out()
```

CPU times: user 4min 35s, sys: 10.1 s, total: 4min 45s
Wall time: 42.1 s

```
[ ]: print("Top terms per cluster:")
print()
#sort cluster centers by proximity to centroid
order_centroids = km.cluster_centers_.argsort()[:, :-1] #sort in descending
↪order

for i in range(num_clusters):
    print("Cluster %d words:" % i, end='')

    for ind in order_centroids[i, :15]: # with 15 words per cluster
        print(' %s' % terms[ind],end=',')
    print() #add whitespace

print()
print()
```

Top terms per cluster:

```
Cluster 0 words: data, variabl, model, use, would, sampl, valu, distribut,
probabl, test, hypothesi, observ, mean, studi, differ,
Cluster 1 words: would, use, becaus, one, peopl, differ, think, make, exampl,
also, like, could, system, way, argument,
Cluster 2 words: problem, solut, solv, constraint, water, use, rightproblem,
differ, identifi, state, goal, subproblem, breakitdown, step, one,
Cluster 3 words: compani, market, product, custom, risk, busi, com, countri,
https, invest, cost, economi, price, googl, doc,
Cluster 4 words: poll, complet, student, present,
faazillexzvnbceqmeprnivrzaadmewqcnu, fabian, fabianokafor, fabiola, fabl,
fabric, fabul, fac, faabi, facad, facbook,
```

```
[ ]: sns.countplot('Assessment reports Score', hue = 'cluster', data = df)
plt.title('Assessment Score Count by Cluster')
```

/Users/swimmingcircle/Library/Python/3.9/lib/python/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

```
warnings.warn(
```

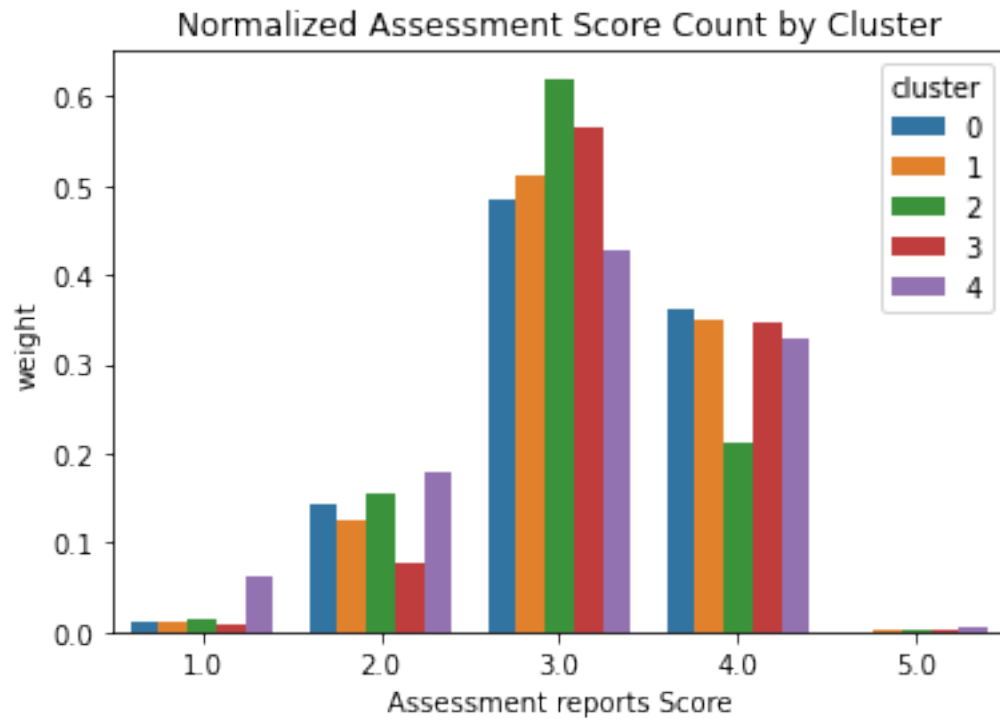
```
[ ]: Text(0.5, 1.0, 'Assessment Score Count by Cluster')
```



```
[ ]: # Normalize the score distribution per cluster
norm_df = df["Assessment reports Score"].groupby(df["cluster"]).value_counts().
    ↪ rename('count').reset_index()
norm_df = norm_df.assign(weight=norm_df['count']/norm_df.
    ↪ groupby('cluster')['count'].transform('sum'))

sns.barplot(x="Assessment reports Score", y="weight", hue = 'cluster',
    ↪ data=norm_df)
plt.title('Normalized Assessment Score Count by Cluster')
```

```
[ ]: Text(0.5, 1.0, 'Normalized Assessment Score Count by Cluster')
```

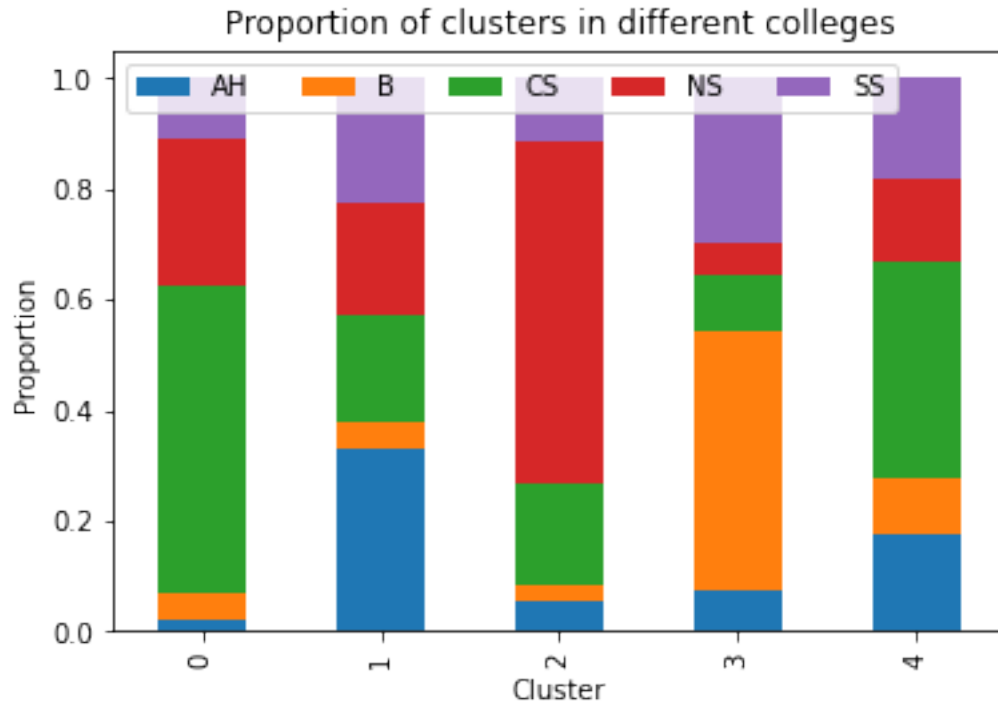


```
[ ]: # College distribution per cluster
cross_tab_prop = pd.crosstab(index=df['cluster'],
                             columns=df['College'],
                             normalize="index")

cross_tab_prop

cross_tab_prop.plot(kind='bar',
                    stacked=True)

plt.legend(loc="upper left", ncol = 5)
plt.xlabel("Cluster")
plt.ylabel("Proportion")
plt.title("Proportion of clusters in different colleges")
plt.show()
```



3.3.2 10 clusters

```
[ ]: km_10 = KMeans(n_clusters=10)

%time km_10.fit(tfidf_matrix)

clusters10 = km_10.labels_.tolist()

df['cluster10'] = np.array(clusters10)
terms10 = tfidf_vector.get_feature_names_out()
```

CPU times: user 6min 34s, sys: 15.1 s, total: 6min 49s

Wall time: 1min 4s

```
[ ]: print("Top terms per cluster:")
print()
#sort cluster centers by proximity to centroid
order_centroids = km_10.cluster_centers_.argsort()[:, :-1]

for i in range(10):
    print("Cluster %d words:" % i, end='')

    for ind in order_centroids[i, :15]: #replace 6 with n words per cluster
        print(' %s' % terms10[ind],end=',')
```



```

print() #add whitespace

print()
print()

```

Top terms per cluster:

Cluster 0 words: doc, https, com, googl, edit, document, usp, share, kgi, edu, minerva, spreadsheet, drive, colab, gid,
Cluster 1 words: would, time, use, function, number, valu, becaus, one, chang, probabl, vector, get, node, first, tree,
Cluster 2 words: system, level, emerg, agent, interact, individu, properti, complex, network, differ, predict, behavior, analysi, social, understand,
Cluster 3 words: variabl, studi, hypothesi, test, treatment, control, observ, group, experi, effect, would, confound, differ, use, research,
Cluster 4 words: compani, market, product, custom, risk, busi, countri, invest, cost, economi, price, would, growth, financi, increas,
Cluster 5 words: poll, complet, student, present, faazillexzvnbceqmeprnivrzaadmewqcnu, fabian, fabianokafor, fabiola, fabl, fabric, fabul, fac, faabi, facad, facbook,
Cluster 6 words: peopl, think, use, one, would, becaus, differ, make, way, like, also, exampl, understand, could, person,
Cluster 7 words: data, model, distribut, sampl, use, probabl, mean, would, valu, predict, line, normal, graph, differ, variabl,
Cluster 8 words: argument, thesi, sentenc, evid, induct, logic, premis, deduct, true, conclus, statement, use, truth, claim, clone,
Cluster 9 words: problem, solut, solv, constraint, use, water, rightproblem, identifi, differ, state, goal, breakitdown, subproblem, step, one,

```

[ ]: sns.countplot('Assessment reports Score', hue = 'cluster10', data = df)
plt.title('Assessment Score Count by Cluster')

```

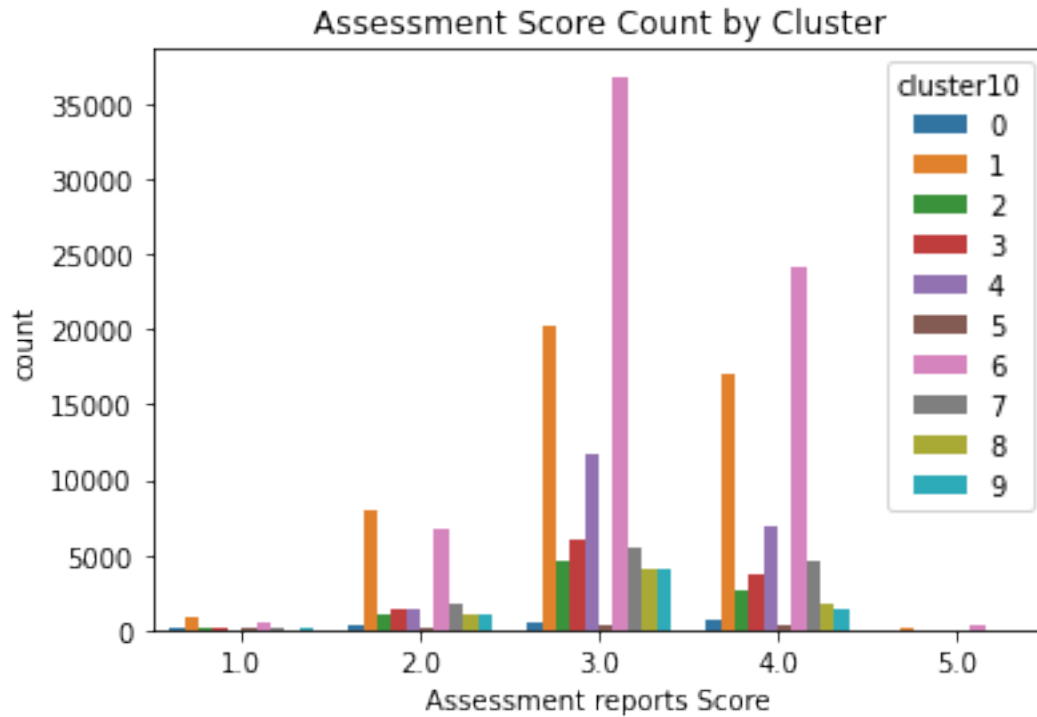
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

```
warnings.warn(
```

```

[ ]: Text(0.5, 1.0, 'Assessment Score Count by Cluster')

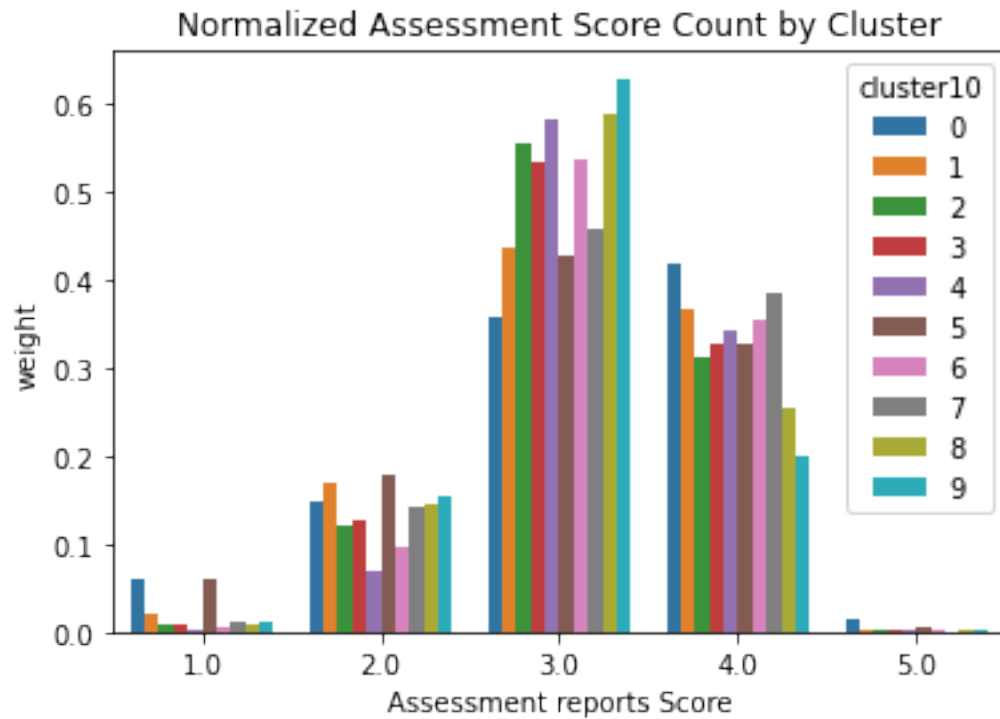
```



```
[ ]: norm_df10 = df["Assessment reports Score"].groupby(df["cluster10"]).
      ↳value_counts().rename('count').reset_index()
norm_df10 = norm_df10.assign(weight=norm_df10['count']/norm_df10.
      ↳groupby('cluster10')['count'].transform('sum'))

sns.barplot(x="Assessment reports Score", y="weight", hue = 'cluster10',
      ↳data=norm_df10)
plt.title('Normalized Assessment Score Count by Cluster')
```

```
[ ]: Text(0.5, 1.0, 'Normalized Assessment Score Count by Cluster')
```

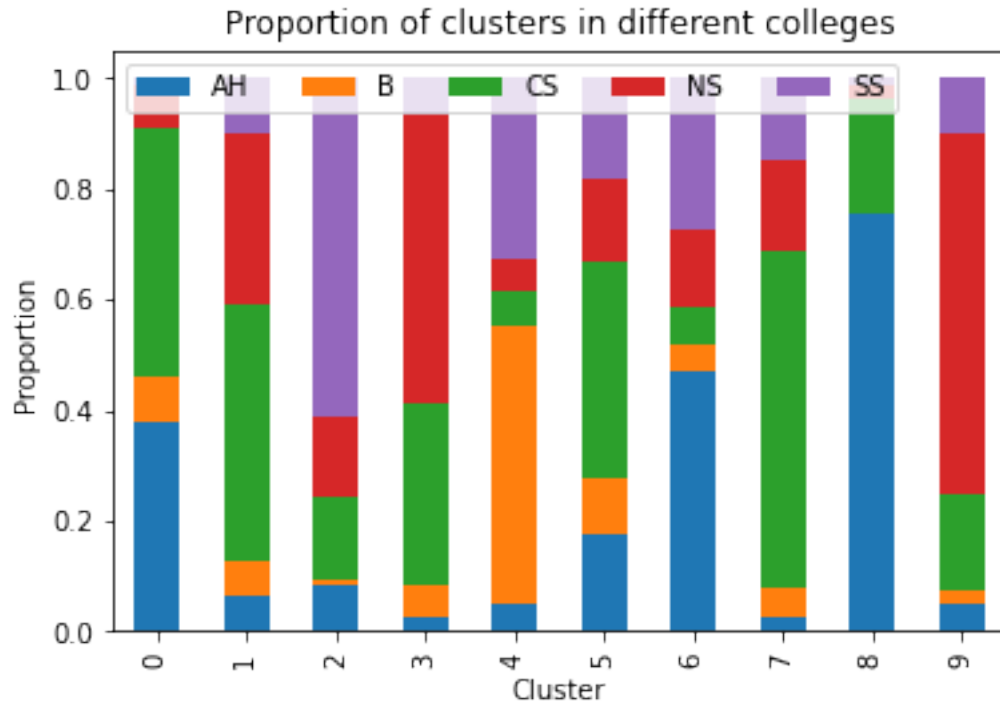


```
[ ]: cross_tab_prop = pd.crosstab(index=df['cluster10'],
                                columns=df['College'],
                                normalize="index")

cross_tab_prop

cross_tab_prop.plot(kind='bar',
                    stacked=True)

plt.legend(loc="upper left", ncol = 5)
plt.xlabel("Cluster")
plt.ylabel("Proportion")
plt.title("Proportion of clusters in different colleges")
plt.show()
```



4 Visualize K-means result

We use TruncatedSVD to decompose `tfidf_matrix`, because the matrix is too fat to plot it directly and TruncatedSVD works well with sparse data. We then attempt to plot the clusters with 2 components.

```
[ ]: from sklearn.decomposition import TruncatedSVD

labels_color_map = {
    0: '#20b2aa', 1: '#ff7373', 2: '#ffe4e1', 3: '#005073', 4: '#4d0404'}

svd = TruncatedSVD(n_components=2)
reduced_matrix = svd.fit_transform(tfidf_matrix)
print(reduced_matrix.shape)
```

```
(181941, 2)
```

```
[ ]: #slice the matrix even more
rr_matrix = reduced_matrix[:3000,:]
```

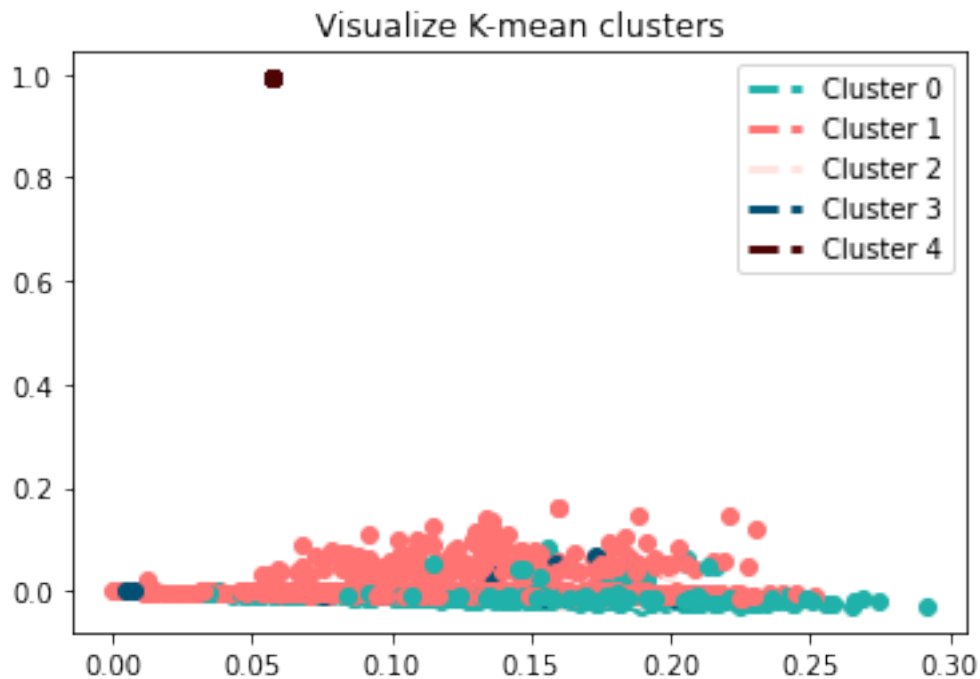
```
[ ]: from matplotlib.lines import Line2D

fig, ax = plt.subplots()
```

```

lines = [Line2D([0], [0], color=c, linewidth=3, linestyle='--') for c in
    ↪labels_color_map.values()]
for index, instance in enumerate(rr_matrix):
    pca_comp_1, pca_comp_2 = rr_matrix[index]
    color = labels_color_map[clusters[index]]
    ax.scatter(pca_comp_1, pca_comp_2, c=color)
plt.title('Visualize K-mean clusters')
plt.legend(lines, ['Cluster 0', 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster_
    ↪4'])
plt.show()

```



However, since the reduced matrix after `TruncatedSVD` is still too big to plot, we slice it into 3000 data points. Therefore, some clusters are not included. From the visualization, it seems that the data isn't accurately clustered. However, it can be because of over reduced and slicing of it, or 2 components aren't the right numbers of dimensions to visualize it.

LDA

April 23, 2022

1 Topic Modeling: Latent Dirichlet Allocation (LDA)

Topic modeling is a type of statistical modeling for discovering the abstract “topics” that occur in a collection of documents, and Latent Dirichlet Allocation (LDA) is one of the method. LDA is a generative probabilistic model that assumes each topic is a mixture over an underlying set of words, and each document is a mixture of over a set of topic probabilities.

Main idea - Every documents is a mixture of topics. e.g. Document 1 is 90% topic A and 10% topic B, while Document 2 is 30% topic A and 70% topic B.

- Every topic is a mixture of words. e.g. two-topic model of American news, with one topic for “politics” and one for “entertainment.” The most common words in the politics topic might be “President”, “Congress”, and “government”, while the entertainment topic may be made up of words such as “movies”, “television”, and “actor”.

Three main parts of LDA

1. Dimensionality Reduction: Rather than representing a text T in its feature space as $\{\text{Word}_i: \text{count}(\text{Word}_i, T) \text{ for } \text{Word}_i \text{ in Vocabulary}\}$, you can represent it in a topic space as $\{\text{Topic}_i: \text{Weight}(\text{Topic}_i, T) \text{ for } \text{Topic}_i \text{ in Topics}\}$.
2. Unsupervised Learning: By doing topic modeling, we build clusters of words rather than clusters of texts. A text is thus a mixture of all the topics, each having a specific weight.
3. Tagging: abstract “topics” that occur in a collection of documents that best represents the information in them.

- ψ , the distribution of words for each topic K
- ϕ , the distribution of topics for each document i
- α : parameter is Dirichlet prior concentration parameter that represents document-topic density — with a higher alpha, documents are assumed to be made up of more topics and result in more specific topic distribution per document.
- β : is the same prior concentration parameter that represents topic-word density — with high beta, topics are assumed to made of up most of the words and result in a more specific word distribution per topic.

2 Import data

```
[ ]: import numpy as np
import pandas as pd
import sklearn
import seaborn as sns
import matplotlib.pyplot as plt
```

```
[ ]: df = pd.read_csv('processed_response.zip')
```

- what is lambda?
- how to interpret the result of the topics?

2.1 Running LDA using bags of words

```
[ ]: import gensim
import gensim.corpora as corpora
```

```
[ ]: from ast import literal_eval

#convert the response to a list
df['clean_responses'] = df['clean_responses'].apply(literal_eval)
```

```
[ ]: # Create Dictionary
id2word = corpora.Dictionary(df['clean_responses'])

# Create Corpus
texts = df['clean_responses']

# Term Document Frequency
corpus = [id2word.doc2bow(text) for text in texts]

print(corpus[0])
```

```
[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 2), (8, 1), (9, 1),
(10, 1), (11, 1), (12, 1), (13, 1), (14, 1), (15, 1), (16, 1), (17, 1), (18, 1),
(19, 1), (20, 1), (21, 1), (22, 1), (23, 1)]
```

```
[ ]: #times a word appears
corpus_0 = corpus[0]
for i in range(len(corpus_0)):
    print("Word {} ({}) appears {} time.".format(corpus_0[i][0],
↪id2word[corpus_0[i][0]], corpus_0[i][1]))
```

Word 0 ("abstract") appears 1 time.
Word 1 ("answer") appears 1 time.
Word 2 ("approach") appears 1 time.
Word 3 ("attack") appears 1 time.
Word 4 ("construct") appears 1 time.

In **corpus**, every tuple represent (index of the word, frequency), which is shown above.

```
[('0,
    '0.013*"would" + 0.011*"becaus" + 0.009*"one" + 0.007*"use" + 0.007*"make" + '
    '0.006*"like" + 0.006*"chang" + 0.006*"differ" + 0.005*"data" + '
    '0.005*"increas"''),
(1,
    '0.016*"system" + 0.011*"would" + 0.008*"level" + 0.007*"think" + '
    '0.007*"individu" + 0.007*"understand" + 0.007*"peopl" + 0.007*"one" + '
    '0.007*"differ" + 0.006*"exempl"''),
(2,
    '0.012*"would" + 0.011*"model" + 0.009*"one" + 0.009*"use" + 0.009*"valu" + '
    '0.008*"n" + 0.008*"p" + 0.008*"time" + 0.008*"because" + 0.007*"differ"''),
(3,
    '0.014*"use" + 0.013*"x" + 0.010*"studi" + 0.009*"would" + 0.009*"one" + '
    '0.008*"because" + 0.007*"research" + 0.007*"understand" + 0.007*"also" + '
    '0.007*"think"''),
(4,
    '0.010*"would" + 0.010*"use" + 0.009*"could" + 0.009*"product" + '
```



```

'0.007*"like" + 0.006*"becaus" + 0.006*"risk" + 0.006*"also" + '
'0.006*"market" + 0.006*"compani"'),
(5,
'0.011*"use" + 0.008*"becaus" + 0.007*"state" + 0.007*"would" + '
'0.006*"differ" + 0.006*"think" + 0.005*"problem" + 0.005*"chang" + '
'0.005*"peopl" + 0.005*"exampl"'),
(6,
'0.018*"use" + 0.011*"data" + 0.009*"variabl" + 0.008*"make" + '
'0.007*"becaus" + 0.007*"line" + 0.007*"would" + 0.006*"one" + 0.006*"show" '
'+ 0.006*"exampl"'),
(7,
'0.010*"would" + 0.010*"compani" + 0.008*"differ" + 0.007*"peopl" + '
'0.007*"work" + 0.006*"time" + 0.006*"use" + 0.005*"one" + 0.005*"becaus" + '
'0.005*"exampl"'),
(8,
'0.012*"peopl" + 0.009*"one" + 0.009*"theori" + 0.007*"becaus" + '
'0.006*"level" + 0.006*"make" + 0.005*"differ" + 0.005*"exampl" + '
'0.005*"polit" + 0.005*"social"'),
(9,
'0.012*"argument" + 0.012*"problem" + 0.011*"use" + 0.011*"evid" + '
'0.009*"make" + 0.008*"becaus" + 0.007*"would" + 0.007*"think" + '
'0.007*"bias" + 0.006*"exampl"')]

```

From the Topic 0 is a represented as 0.013 “would” + 0.011 “becaus” + 0.009 “one” + 0.007 “use” + 0.007 “make” + 0.006 “like” + 0.006 “chang” + 0.006 “differ” + 0.005 “data” + 0.005 “increas.” It means the top 10 keywords that contribute to this topic are: would, because, one.. and so on and the weight of would on topic 0 is 0.013.

```

[ ]: #print the top 20 words in each topics
topics_matrix = lda_model.show_topics(formatted=False, num_words=20)
topics_matrix = np.array(topics_matrix)
topic_words = topics_matrix[:,1]

for i in topic_words:
    print([str(word[0]) for word in i])
    print()

```

```

['would', 'becaus', 'one', 'use', 'make', 'like', 'chang', 'differ', 'data',
'increas', 'mean', 'exampl', 'group', 'c', 'also', 'need', 'think', 'v',
'activ', 'water']

```

```

['system', 'would', 'level', 'think', 'individu', 'understand', 'peopl', 'one',
'differ', 'exampl', 'effect', 'complex', 'need', 'interact', 'becaus', 'also',
'group', 'social', 'market', 'motiv']

```

```

['would', 'model', 'one', 'use', 'valu', 'n', 'p', 'time', 'becaus', 'differ',
'number', 'probabl', 'b', 'distribut', 'x', 'function', 'first', 'data', 'case',
'also']

```

```
['use', 'x', 'studi', 'would', 'one', 'becaus', 'research', 'understand',  
'also', 'think', 'differ', 'effect', 'need', 'help', 'way', 'could', 'make',  
'hypothesi', 'test', 'data']
```

```
['would', 'use', 'could', 'product', 'like', 'becaus', 'risk', 'also', 'market',  
'compani', 'think', 'one', 'us', 'creat', 'make', 'invest', 'differ', 'countri',  
'increas', 'idea']
```

```
['use', 'becaus', 'state', 'would', 'differ', 'think', 'problem', 'chang',  
'peopl', 'exampl', 'one', 'way', 'mean', 'make', 'also', 'class', 'hypothesi',  
'could', 'structur', 'network']
```

```
['use', 'data', 'variabl', 'make', 'becaus', 'line', 'would', 'one', 'show',  
'exampl', 'activ', 'mean', 'bias', 'word', 'point', 'could', 'like', 'control',  
'also', 'differ']
```

```
['would', 'compani', 'differ', 'peopl', 'work', 'time', 'use', 'one', 'becaus',  
'exampl', 'like', 'also', 'make', 'could', 'valu', 'interest', 'way', 'market',  
'help', 'chang']
```

```
['peopl', 'one', 'theori', 'becaus', 'level', 'make', 'differ', 'exampl',  
'polit', 'social', 'would', 'cultur', 'societi', 'human', 'person', 'individu',  
'way', 'base', 'govern', 'moral']
```

```
['argument', 'problem', 'use', 'evid', 'make', 'becaus', 'would', 'think',  
'bias', 'exampl', 'state', 'also', 'thesi', 'one', 'differ', 'way', 'claim',  
'solut', 'base', 'effect']
```

```
/var/folders/0h/xyv81g2n7sj6zr0c9cw30gkc0000gn/T/ipykernel_13803/4262890021.py:3  
: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences  
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths  
or shapes) is deprecated. If you meant to do this, you must specify  
'dtype=object' when creating the ndarray.  
topics_matrix = np.array(topics_matrix)
```

```
[ ]: #visualize the result  
  
import pyLDAvis  
# import pyLDAvis.gensim  
import pyLDAvis.gensim_models as gensimvis  
import os  
import pickle  
  
# Visualize the topics  
pyLDAvis.enable_notebook()
```

```
[ ]: LDAvis_prepared = gensimvis.prepare(lda_model, corpus, id2word)
```

```

    if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
    other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
    if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
    other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
    if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
    other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
    if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
    other = LooseVersion(other)

```

```
[ ]: pyLDavis.display(LDAvis_prepared)
```

```
[ ]: <IPython.core.display.HTML object>
```

The relevance metric: λ

- $\lambda = 1$: ranking solely by probability in descending order
- $\lambda = 0$: ranking solely by the lift. lift = the ratio of a term's probability within a topic to its marginal probability across the corpus. It generally decreases with globally frequent terms. But it can be noisy too if the rare terms only occur in one single topic.

$P(T|w)$: the likelihood that observed word w was generated by latent topic T . We use it to determine how informative a specific word w can inform the topics. If word tells little of the topic mixture it will receive a low distinctiveness score.

```
[ ]: pyLDavis.save_html(LDAvis_prepared, 'lda.html')
```

2.2 Running LDA using tf-idf

```
[ ]: from gensim import corpora, models
```

```
tfidf = models.TfidfModel(corpus)
corpus_tfidf = tfidf[corpus]
```

```
# Build LDA model on tf-idf
```

```
lda_model_tf = gensim.models.LdaMulticore(corpus=corpus_tfidf,
                                           id2word=id2word,
                                           num_topics=num_topics)
```

```
# Print the Keyword in the 10 topics
```

```
pprint(lda_model_tf.print_topics())
```

```
[(0,
  '0.004*"system" + 0.004*"peopl" + 0.003*"differ" + 0.003*"level" + '
  '0.003*"one" + 0.003*"emot" + 0.003*"would" + 0.003*"think" + '
  '0.003*"understand" + 0.003*"interact"'),
 (1,
  '0.009*"problem" + 0.005*"solut" + 0.004*"solv" + 0.004*"use" + '
  '0.003*"think" + 0.003*"constraint" + 0.003*"would" + 0.003*"differ" + '
  '0.003*"goal" + 0.003*"help"'),
 (2,
  '0.009*"x" + 0.006*"data" + 0.006*"variabl" + 0.006*"distribut" + '
  '0.006*"sampl" + 0.005*"probabl" + 0.005*"valu" + 0.005*"model" + 0.004*"p" '
  '+ 0.004*"would"'),
 (3,
  '0.003*"would" + 0.003*"compani" + 0.003*"peopl" + 0.003*"market" + '
  '0.003*"differ" + 0.003*"think" + 0.002*"make" + 0.002*"one" + 0.002*"use" + '
  '0.002*"need"'),
 (4,
  '0.009*"poll" + 0.009*"student" + 0.008*"complet" + 0.008*"https" + '
  '0.007*"present" + 0.007*"com" + 0.007*"googl" + 0.007*"doc" + 0.006*"edit" '
  '+ 0.006*"document"'),
 (5,
  '0.007*"n" + 0.004*"would" + 0.004*"node" + 0.004*"algorithm" + 0.003*"tree" '
  '+ 0.003*"time" + 0.003*"number" + 0.003*"use" + 0.003*"one" + 0.003*"list"'),
 (6,
  '0.003*"would" + 0.002*"co" + 0.002*"x" + 0.002*"increas" + 0.002*"water" + '
  '0.002*"becaus" + 0.002*"chang" + 0.002*"system" + 0.002*"use" + '
  '0.002*"peopl"'),
 (7,
  '0.006*"compani" + 0.004*"market" + 0.003*"product" + 0.003*"would" + '
  '0.003*"use" + 0.003*"custom" + 0.003*"risk" + 0.003*"cost" + 0.003*"make" + '
  '0.002*"think"'),
 (8,
```

```
'0.003*"would" + 0.003*"use" + 0.003*"energi" + 0.002*"effect" + '
'0.002*"becaus" + 0.002*"differ" + 0.002*"water" + 0.002*"one" + '
'0.002*"cell" + 0.002*"observ"'),
(9,
'0.007*"argument" + 0.006*"evid" + 0.005*"p" + 0.005*"hypothesi" + 0.004*"b" '
'+ 0.004*"induct" + 0.004*"thesi" + 0.004*"c" + 0.004*"true" + 0.004*"use"'))]
```

```
[ ]: #print the top 20 words in each topics
topics_matrix = lda_model_tf.show_topics(formatted=False, num_words=20)
topics_matrix = np.array(topics_matrix)
topic_words = topics_matrix[:,1]

for i in topic_words:
    print([str(word[0]) for word in i])
    print()
```

```
['system', 'peopl', 'differ', 'level', 'one', 'emot', 'would', 'think',
'understand', 'interact', 'use', 'individu', 'cultur', 'becaus', 'agent',
'social', 'way', 'exampl', 'emerg', 'make']
```

```
['problem', 'solut', 'solv', 'use', 'think', 'constraint', 'would', 'differ',
'goal', 'help', 'one', 'process', 'understand', 'need', 'appli', 'level',
'becaus', 'could', 'identifi', 'activ']
```

```
['x', 'data', 'variabl', 'distribut', 'sampl', 'probabl', 'valu', 'model', 'p',
'would', 'mean', 'use', 'function', 'treatment', 'vector', 'calcul', 'differ',
'number', 'test', 'random']
```

```
['would', 'compani', 'peopl', 'market', 'differ', 'think', 'make', 'one', 'use',
'need', 'strategi', 'becaus', 'system', 'could', 'product', 'countri', 'also',
'chang', 'effect', 'group']
```

```
['poll', 'student', 'complet', 'https', 'present', 'com', 'googl', 'doc',
'edit', 'document', 'usp', 'share', 'would', 'use', 'argument', 'think',
'becaus', 'one', 'make', 'could']
```

```
['n', 'would', 'node', 'algorithm', 'tree', 'time', 'number', 'use', 'one',
'list', 'becaus', 'valu', 'sort', 'first', 'make', 'x', 'optim', 'case', 'row',
'think']
```

```
['would', 'co', 'x', 'increas', 'water', 'becaus', 'chang', 'system', 'use',
'peopl', 'one', 'differ', 'carbon', 'time', 'histogram', 'earth', 'state',
'temperatur', 'citi', 'could']
```

```
['compani', 'market', 'product', 'would', 'use', 'custom', 'risk', 'cost',
'make', 'think', 'peopl', 'invest', 'price', 'activ', 'rate', 'differ',
'financi', 'also', 'becaus', 'busi']
```

```
['would', 'use', 'energi', 'effect', 'becaus', 'differ', 'water', 'one', 'cell',  
'observ', 'test', 'could', 'chang', 'hypothesi', 'increas', 'complianc', 'like',  
'theori', 'time', 'level']
```

```
['argument', 'evid', 'p', 'hypothesi', 'b', 'induct', 'thesi', 'c', 'true',  
'use', 'data', 'conclus', 'valid', 'premis', 'deduct', 'sentenc', 'logic',  
'theori', 'test', 'q']
```

```
/var/folders/0h/xyv81g2n7sj6zr0c9cw30gkc0000gn/T/ipykernel_13803/697313039.py:3:  
VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences  
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths  
or shapes) is deprecated. If you meant to do this, you must specify  
'dtype=object' when creating the ndarray.  
topics_matrix = np.array(topics_matrix)
```

```
[ ]: LDAvis_prepared_tf = gensimvis.prepare(lda_model_tf, corpus_tfidf, id2word)
```

```
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-  
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes  
are deprecated. Use packaging.version instead.  
    if LooseVersion(np.__version__) < '1.13':  
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-  
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils  
Version classes are deprecated. Use packaging.version instead.  
    other = LooseVersion(other)  
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-  
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes  
are deprecated. Use packaging.version instead.  
    if LooseVersion(np.__version__) < '1.13':  
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-  
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils  
Version classes are deprecated. Use packaging.version instead.  
    other = LooseVersion(other)  
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-  
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes  
are deprecated. Use packaging.version instead.  
    if LooseVersion(np.__version__) < '1.13':  
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-  
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils  
Version classes are deprecated. Use packaging.version instead.  
    other = LooseVersion(other)  
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-  
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes  
are deprecated. Use packaging.version instead.  
    if LooseVersion(np.__version__) < '1.13':  
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-  
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes  
are deprecated. Use packaging.version instead.
```

```

    other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
    if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
    other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
    if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
    other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
    if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
    other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
    if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
    other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
    if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
    other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
    if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.

```

```

other = LooseVersion(other)

[ ]: pyLDavis.display(LDAvis_prepared_tf)

[ ]: <IPython.core.display.HTML object>

[ ]: pyLDavis.save_html(LDAvis_prepared_tf, 'lda_tf.html')

```

3 Select the best number of clusters by coherence score

We can coherence score in topic modeling to measure how interpretable the topics are to humans. We select the best number of clusters based on highest coherence score.

```

[ ]: from gensim.models import CoherenceModel

# Compute Coherence Score
coherence_model_lda = CoherenceModel(model=lda_model, texts=texts,
    ↪dictionary=id2word, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print('Coherence Score bag of words: ', coherence_lda)

coherence_model_lda = CoherenceModel(model=lda_model_tf, texts=texts,
    ↪dictionary=id2word, coherence='c_v')
coherence_lda_tf = coherence_model_lda.get_coherence()
print('Coherence Score tf-idf: ', coherence_lda_tf)

```

```

Coherence Score bag of words: 0.33813657622075743
Coherence Score tf-idf: 0.3884673157891732

```

```

[ ]: num_list = list(range(2,11))
coherence = {}
for num_topics in num_list:
    lda_model_tf = gensim.models.LdaMulticore(corpus=corpus_tfidf,
        id2word=id2word,
        num_topics=num_topics)
    coherence_model_lda = CoherenceModel(model=lda_model_tf, texts=texts,
    ↪dictionary=id2word, coherence='c_v')
    coherence_lda = coherence_model_lda.get_coherence()
    coherence[num_topics] = coherence_lda

```

```

[ ]: coherence

```

```

[ ]: {2: 0.29721896496192934,
3: 0.31396921106978004,
4: 0.34535485103863095,
5: 0.2559763428291707,
6: 0.3195102597091959,
7: 0.3465651203928427,

```



```

8: 0.3767523695290471,
9: 0.3953096197373716,
10: 0.38321805318943086}

```

The next can be tuning the hyperparameters for alpha and beta. - [Evaluate Topic Models: Latent Dirichlet Allocation \(LDA\)](#)

```

[ ]: n = 9

# Build LDA model on tf-idf
lda_model_tf = gensim.models.LdaMulticore(corpus=corpus_tfidf,
                                           id2word=id2word,
                                           num_topics= n)

# Print the Keyword in the 10 topics
pprint(lda_model_tf.print_topics())

[(0,
  '0.004*"data" + 0.004*"use" + 0.004*"studi" + 0.003*"would" + 0.003*"think" '
  '+ 0.003*"make" + 0.003*"one" + 0.003*"differ" + 0.003*"bias" + '
  '0.003*"audienc"'),
 (1,
  '0.004*"art" + 0.003*"music" + 0.003*"moral" + 0.002*"use" + 0.002*"cultur" '
  '+ 0.002*"would" + 0.002*"peopl" + 0.002*"think" + 0.002*"differ" + '
  '0.002*"one"'),
 (2,
  '0.004*"variabl" + 0.004*"attent" + 0.004*"memori" + 0.004*"line" + '
  '0.004*"data" + 0.003*"slope" + 0.003*"regress" + 0.003*"would" + '
  '0.003*"process" + 0.003*"correl"'),
 (3,
  '0.006*"system" + 0.005*"problem" + 0.004*"model" + 0.003*"level" + '
  '0.003*"differ" + 0.003*"agent" + 0.003*"use" + 0.003*"interact" + '
  '0.003*"complex" + 0.003*"solut"'),
 (4,
  '0.013*"x" + 0.009*"n" + 0.007*"p" + 0.005*"b" + 0.005*"valu" + '
  '0.005*"function" + 0.005*"number" + 0.004*"tree" + 0.004*"f" + '
  '0.004*"probabl"'),
 (5,
  '0.006*"student" + 0.005*"poll" + 0.004*"complet" + 0.004*"present" + '
  '0.003*"peopl" + 0.003*"would" + 0.003*"countri" + 0.002*"differ" + '
  '0.002*"becaus" + 0.002*"use"'),
 (6,
  '0.004*"peopl" + 0.003*"leader" + 0.003*"would" + 0.003*"one" + '
  '0.003*"state" + 0.003*"power" + 0.002*"system" + 0.002*"individu" + '
  '0.002*"becaus" + 0.002*"think"'),
 (7,
  '0.005*"thesi" + 0.005*"argument" + 0.004*"evid" + 0.004*"use" + 0.003*"flu" '
  '+ 0.003*"clone" + 0.003*"would" + 0.002*"make" + 0.002*"effect" + '
  '0.002*"becaus"'),

```

```
(8,
 '0.007*"compani" + 0.006*"market" + 0.005*"product" + 0.004*"custom" + '
 '0.004*"risk" + 0.004*"valu" + 0.004*"cost" + 0.004*"would" + 0.004*"price" '
 '+ 0.003*"rate"'))]
```

```
[ ]: #print the top 20 words in each topics
topics_matrix = lda_model_tf.show_topics(formatted=False, num_words=20)
topics_matrix = np.array(topics_matrix)
topic_words = topics_matrix[:,1]

for i in topic_words:
    print([str(word[0]) for word in i])
    print()
```

```
['data', 'use', 'studi', 'would', 'think', 'make', 'one', 'differ', 'bias',
'audienc', 'hypothesi', 'research', 'could', 'understand', 'googl', 'design',
'variabl', 'effect', 'becaus', 'test']
```

```
['art', 'music', 'moral', 'use', 'cultur', 'would', 'peopl', 'think', 'differ',
'one', 'becaus', 'also', 'way', 'like', 'work', 'make', 'artist', 'understand',
'context', 'could']
```

```
['variabl', 'attent', 'memori', 'line', 'data', 'slope', 'regress', 'would',
'process', 'correl', 'use', 'brain', 'r', 'model', 'temperatur', 'co',
'increas', 'one', 'becaus', 'differ']
```

```
['system', 'problem', 'model', 'level', 'differ', 'agent', 'use', 'interact',
'complex', 'solut', 'would', 'one', 'emerg', 'solv', 'properti', 'becaus',
'understand', 'think', 'could', 'state']
```

```
['x', 'n', 'p', 'b', 'valu', 'function', 'number', 'tree', 'f', 'probabl',
'distribut', 'algorithm', 'vector', 'c', 'node', 'would', 'matrix', 'z', 'use',
'time']
```

```
['student', 'poll', 'complet', 'present', 'peopl', 'would', 'countri', 'differ',
'becaus', 'use', 'think', 'one', 'social', 'chang', 'econom', 'like', 'could',
'class', 'effect', 'exempl']
```

```
['peopl', 'leader', 'would', 'one', 'state', 'power', 'system', 'individu',
'becaus', 'think', 'level', 'differ', 'polit', 'group', 'govern', 'social',
'chang', 'make', 'exempl', 'could']
```

```
['thesi', 'argument', 'evid', 'use', 'flu', 'clone', 'would', 'make', 'effect',
'becaus', 'word', 'one', 'exempl', 'organ', 'think', 'support', 'statement',
'claim', 'c', 'reader']
```

```
['compani', 'market', 'product', 'custom', 'risk', 'valu', 'cost', 'would',
'price', 'rate', 'busi', 'invest', 'increas', 'financi', 'strategi', 'growth',
```

```
'differ', 'because', 'need', 'brand']
```

```
/var/folders/0h/xyv81g2n7sj6zr0c9cw30gkc0000gn/T/ipykernel_13803/697313039.py:3:
VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify
'dtype=object' when creating the ndarray.
    topics_matrix = np.array(topics_matrix)
```

```
[ ]: LDAvis_prepared_tf = gensimvis.prepare(lda_model_tf, corpus_tfidf, id2word)
```

```
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/pyLDAvis/_prepare.py:246: FutureWarning: In a future version of pandas
all arguments of DataFrame.drop except for the argument 'labels' will be
keyword-only
```

```
    default_term_info = default_term_info.sort_values(
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is
deprecated in favour of importlib; see the module's documentation for
alternative uses
```

```
    from imp import reload
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is
deprecated in favour of importlib; see the module's documentation for
alternative uses
```

```
    from imp import reload
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is
deprecated in favour of importlib; see the module's documentation for
alternative uses
```

```
    from imp import reload
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is
deprecated in favour of importlib; see the module's documentation for
alternative uses
```

```
    from imp import reload
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is
deprecated in favour of importlib; see the module's documentation for
alternative uses
```

```
    from imp import reload
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is
deprecated in favour of importlib; see the module's documentation for
alternative uses
```

```
    from imp import reload
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is
```

```

/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
    if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
    other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
    if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
    other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
    if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
    other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
    if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
    other = LooseVersion(other)

```

```

[ ]: pyLDAvis.display(LDAvis_prepared_tf)
     pyLDAvis.save_html(LDAvis_prepared_tf, '9_lda_tf.html')

```