Get it off your chest --- What topics do most people confess about?

Problem Statement:

In this project, I will compare these 3 techniques: K-means clustering, Latent Semantic Analysis and Latent Dirichlet Allocation (LDA), to get the hidden topics of each data point, and explore clustering methods to analyze what topics most people confess about on Reddit.

This is an unsupervised learning project, so we do not have true cluster labels. In each model, I will tune parameters to find the optimal number of topics and interpret each topic. The evaluation methods are topic coherence values for LDA and LSA, and silhouette coefficient scores for K-means clustering.

I also discuss about the challenges and solutions when dealing with a million data, such as mini-batches and online learning are helpful to speed up the process while not taking too much memory. LDA method works the best to extract hidden topics among the three topics, based on the coherence values. It extracts 18 topics, covering from work, family, school, friends, relationships, and many other aspects. What can be done to make improvement is that there are other parameters can be tuned, different evaluation methods can be used, and more topic modelling models that can be used, to test and improve the goodness of fit for the data.

Data Source:

Link: [One Million Reddit Confessions | Kaggle]

Raw Data example: There are about 1 million raw data before cleaning. Each row represents a post on Reddit. I will focus on the "selftext" and "title" column, as it is the main text of each confession. In the implementation, I put "title" and "selftext" columns together to treat them as one document.

type	id	subredd	subredd	subredd	created_	permali	domain	url	selftext	title	score
post	pyw04g	2qo2a	confession	FALSE	1.63E+09	https://old	self.confession			I was horril	8
post	pyvwgp	2qo2a	confession	FALSE	1.63E+09	https://old	self.confession		Hello	I have 0 frie	22
post	pyvj28	2qo2a	confession	FALSE	1.63E+09	https://old	self.confession		My son is 2	I set up can	0
post	pytdpo	2qo2a	confession	FALSE	1.63E+09	https://old	self.confession		I did one	I drove whi	0
post	pyr64d	2qo2a	confession	FALSE	1.63E+09	https://old	self.confession		So for a lon	I stole $200	3
post	pyqs8t	2qo2a	confession	FALSE	1.63E+09	https://old	self.confession		I was a teer	I took some	2
post	pynmzd	2qo2a	confession	FALSE	1.63E+09	https://old	self.confession		In the	In the mid !	796
post	pyc5uv	2qo2a	confession	FALSE	1.63E+09	https://old	self.confession		When I was	Took a mar	14
post	py5vsy	2qo2a	confession	FALSE	1.63E+09	https://old	self.trueoff	/r/TrueOffMyChest/cc		I hate how	1
post	pw87ib	2qo2a	confession	FALSE	1.63E+09	https://old	self.confession		so you	i tried to te	2449
post	pvml3s	2qo2a	confession	FALSE	1.63E+09	https://old	self.confession		When I was	I stole from	310

Methodology:

1. Data Preprocessing:
   a. Remove empty or uninformative posts.
   b. Ignore case.
   c. Ignore punctuation, emojis, and special characters.
   d. Remove shorter words, such as "a", "the", "am", and stop words, such as "edit", "hello", which are common in reddit contents.

e. Lemmatize words, such as “run”, “ran”, ”running” stem from “run”, so they should be treated as the same word. Lemmatize and Stem words are different. Stem will physically keep roots of similar words, such as “having” and “haves” will become “hav”. However, “broke” and “break” are treated as different words. I use lemmatization because we want to understand and interpret the topics easily, as we want the hidden meanings from each topic.

2. K-means clustering, Latent Semantic Analysis or Latent Dirichlet Allocation?

Now, we have the cleaned data. The next question is, which technique is better for extract the topics in the text? To start, we need to understand the meaning of “topic”. A topic, mathematically, is a distribution of words. Topic modeling is, essentially, a text clustering problem. In the project, I consider three different kinds of text clustering technique: K-means, Latent Semantic Analysis and Latent Dirichlet Allocation.

a) K-means clustering:

Clustering is one of the most used unsupervised machine learning algorithms. K-Means is one of the simplest clustering algorithms to detect common patterns in unorganized data points. The algorithm classifies all data points into K clusters by identifying close data points based on their similarities. K-Means simply defines similarity as the Euclidean distance measured in the feature space. To cluster text data, the first thing to do is to transform text into quantified form.

The cleaned data is converted into a quantified matrix, where each document represents a vector. In each vector, there will be words and its frequency. Bag of Words (BoW) model can represent text in numbers. The matrix generated by bag of words model is a $m \times n$ matrix, where m is the number of document and n is the number of unique words. However, the drawback of bag of words model is that it can be grow very large quickly as the number of vocabularies increases, and uninformative but common words will be retrieved as results.

A better method is Tf-idf vectorizer: Tf-idf stands for term frequency-inverse document frequency. Alter the document term matrix by applying Tf-idf method, to evaluate how important a word is to a collection. Tf-idf value is directly proportional to the number of appearances of a word in a document and is offset by the number of documents containing that word, which compensates for the fact that some words appear more commonly than others in general. In Tf-idf model, each document and term have its own TF value. But just computing the TF alone is not sufficient to understand the importance of words. So IDF value for each word shows the importance. In the process, words like “is”, “and”, “this” have not much information will have low importance values.

$$TF(t) = \frac{\text{Number of times term t appears in a document}}{\text{Total number of terms in the document}}$$

$$IDF(t) = log\frac{\text{Total number of documents}}{\text{Number of documents with term t in it}}$$

$$Tfidf = TF(t) * IDF(t)$$

After building the vectorized matrix, now it is time to apply clustering.

i. First, randomly initialize the K starting centroids:$\{c^1, c^2, \dots, c^k\}$. Each document is assigned to its nearest centroid.
ii. The centroids are recomputed as the mean of the data points assigned to the respective cluster.

$$\pi(i) = argmin_{j=1,\dots,k} d\left(x^i, c^j\right)$$

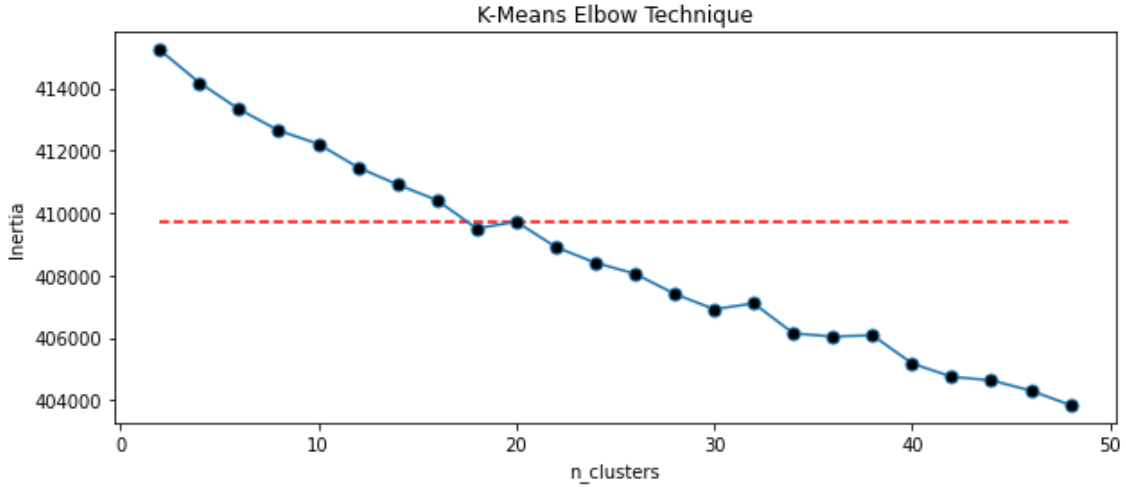
$$c^j = argmin_{v\in\mathbb{R}} \sum_{i:\pi(i)=j} d\left(x^i, v\right)^2$$

iii. Repeat steps 1 and 2 until any cluster center is not changed.

When the number of clusters is too small then result could lose its significances. But too many clusters then results will not generalize well. To solve it and tune it, I will use the elbow method which is a common technique used for this task. It involves estimating the model using various numbers of clusters and calculating the negative of the within-cluster sum of

squares for each number of clusters chosen. The “elbow” point of the graph is usually where the optimal number of clusters is.

Each cluster should represent a topic in the text. Each document only has one topic. The result does not have a smooth elbow method. It indicates that more clusters, the better. But there if there are too many clusters, it could be not meaningful to interpret. If we set the number of topics at 20, samples of different clusters are shown. From the result, we can see that each cluster has a different theme. For example, cluster 8 can be inferred about the stealing and scamming; cluster 3 is about relationships and self-control. But it is also obvious that these clusters have overlapping information and similar words. The overall silhouette score for the model is close to 0, which means it is a mixture of good and bad clustering. A disadvantage of k-means clustering is that each observation, each document—can be assigned to one, and only one, cluster. In real life, documents and sentences can have different themes. For example, “running in the park” can be related to “health” or “activities”. K-means clustering, cannot solve the problem.



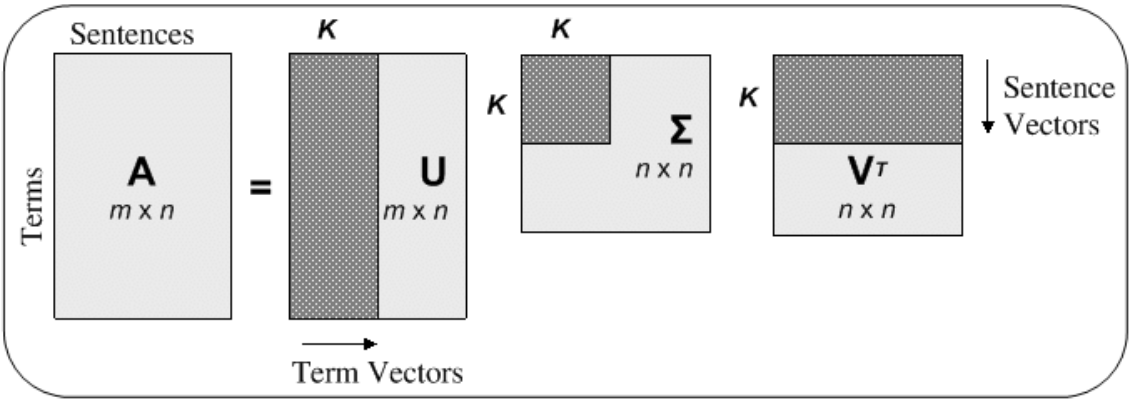
Samples of Clusters:

Cluster	*Sample 1*	*Sample 2*	*Sample 3*	*Sample 4*	*Sample 5*	***Topic***
3	people, hurt, take, hurt, people, nothing	steal, stuff, recently, get, steal, love	drugsalcohol, escape, sorry, anything, spell	today, relapse, months, control, addiction	open, letter, address, happen, minutes, live	**People**
8	steal, money, parent, credit, card, trick	use, collect, people, personal, data	steal, people, steal, account, sell, kind	scammed, 65000, worth, virtual, items, find	modify, late, night, fee, store, keep, money	**Money**
14	steal, little, sister, shop, lie, long, time	kick, hole, drywall, bathroom, things, angry	use, steal, younger, pass, away, younger	join, bully, order, protect, take, place	steal, things, sister, cherish, destroy	**Family Relationship**
19	work, sneak, conference, room, steal, good	spike, entire, offices, coffee, supply	suppose, log, hours, home, today, even, look	steal, 5000, first, become, manager, work	drink, dads, half, half, go, teenager	**Self-control**

b) Latent Semantic Analysis (LSA):

In LSA, word frequencies in the document-term matrix also play a key role in extracting the latent topics. Instead of just assigning each document to one cluster, LSA extracts the dimensions of the text by implementing Singular Value Decomposition (SVD) on the matrix, so that a low-rank approximation of the original matrix can be created.

It works as follows:



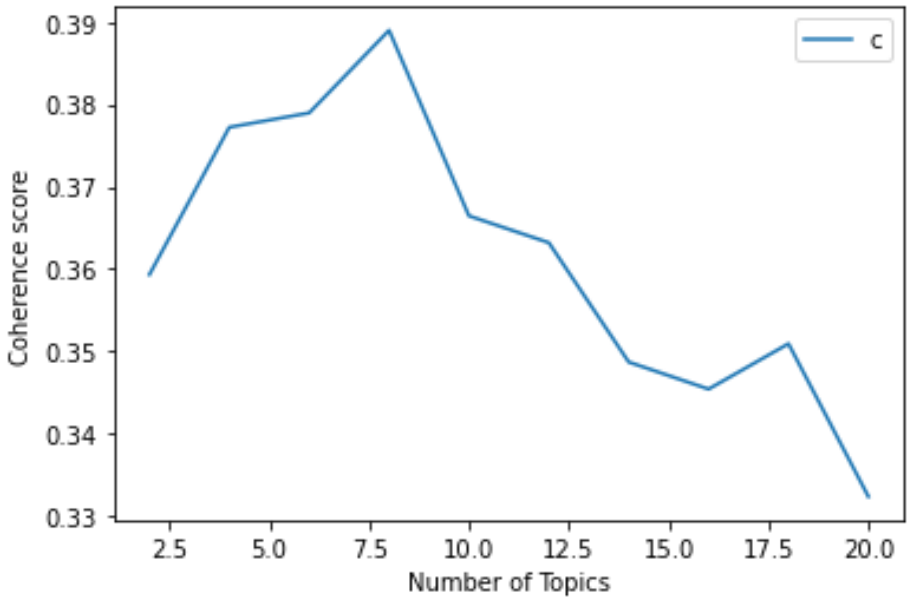
In the decomposed matrix, U is the document-topic matrix. In this matrix, rows represent documents, vectors expressed in terms of topics. V is the term-topic matrix. In this matrix, rows represent terms, vectors expressed in terms of topics. Each row of the matrix U (document-topic matrix) is the vector representation of the corresponding document. Here the length of these vectors is equal to K, which is the number of desired topics we want and vector representation for the terms in our data can be found in the matrix V (term-topic matrix). So, SVD returns vector representations for every document and term in data. The length of each vector would be K. One important use of these vectors is we can find similar words and similar documents with the help of the cosine similarity metric.

After decomposition, LSA measures the similarity between vectors in the reduced dimensional space is the cosine measure: The cosine of the angle between their vectors in the rank-k subspace. The closest documents or all documents exceeding some cosine threshold are returned. The rationale for this is that the vector space into which we embed our documents is defined such that the dimensions in it approximately relate to the concepts within the documents. The vector for a document points in the directions of the concepts that document contains. Therefore, two documents with similar conceptual content will have vectors that point in similar directions: the angle between their vectors will be relatively small, so the cosine of this angle will be larger than that between documents with no conceptual similarity. Values close to 1 represent very similar documents while values close to 0 represent very dissimilar documents.

$$\cos(\theta_{xy}) = \frac{xy}{|x||y|}$$

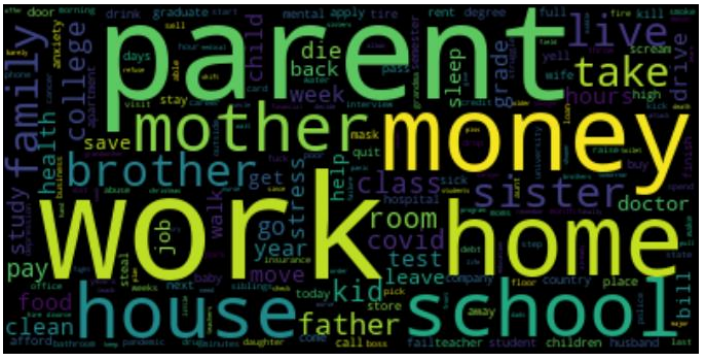
If we plotted these topics and terms in a table, where the rows are the terms, we would see scores plotted for each term according to which topic it most strongly belonged. The higher the number, the greater its affiliation to that topic. They represent how much each of the topics explains our data.

It is also important to decide the number of topics in LSA. One way to determine the optimum number of topics is topic coherence value, it is a realistic measure for identifying the number of topics. In the above image K is the rank of the matrix. It resembles that if we use only k columns and rows, also we can approximately calculate the matrix A without any major loss. LSA gives corpus as a result, it is a matrix comprised of the components from the SVD result. The matrix will give us the latent topics according to the score value. By comparing the score weights on each topic, we can tell which topic the document belongs to. By sorting the absolute value of singular value scores, we can tell the most important words in each topic.



Feelings	• '0.146*"feel" + 0.132*"want" + 0.121*"know" + 0.121*"like" + 0.119*"tell" + 0.119*"love" + 0.110*"time" + 0.110*"friends" + 0.108*"life" + 0.107*"think"'
Relationship	• '-0.291*"love" + -0.215*"friend" + 0.178*"work" + 0.166*"parent" + -0.160*"relationship" + -0.160*"date" + -0.157*"friends" + -0.146*"girl" + -0.145*"talk" + -0.145*"someone"'
Negative feelings	• '0.275*"hate" + -0.253*"friend" + 0.205*"people" + 0.187*"life" + 0.175*"tire" + -0.164*"say" + 0.164*"feel" + -0.150*"tell" + -0.135*"girl" + -0.129*"shes"'
School Life	• '-0.413*"love" + 0.262*"school" + 0.230*"friends" + 0.213*"people" + 0.161*"class" + -0.142*"miss" + -0.134*"mother" + 0.126*"friend" + 0.120*"group" + -0.118*"sister"'

School and relationship	• '0.237*"school" + 0.233*"miss" + -0.182*"hate" + 0.178*"college" + -0.158*"women" + 0.156*"love" + 0.154*"friends" + 0.148*"year" + -0.145*"people" + -0.135*"say"'
Family	• '0.296*"hate" + 0.249*"parent" + 0.242*"school" + 0.218*"sister" + -0.208*"work" + 0.206*"brother" + 0.186*"mother" + 0.184*"friends" + 0.163*"family" + -0.160*"tire"'
Friends	• '-0.300*"tire" + 0.286*"love" + -0.285*"friend" + 0.275*"women" + -0.190*"friends" + 0.160*"date" + -0.154*"birthday" + 0.151*"school" + 0.130*"girl" + -0.125*"talk"'
Money	• '-0.749*"steal" + -0.225*"miss" + -0.183*"money" + -0.131*"store" + -0.124*"love" + -0.111*"card" + -0.101*"hate" + -0.097*"reddit" + -0.096*"birthday" + -0.086*"christmas"'



LSA is efficient and easy to implement. The main advantage of SVD is that we can reduce the size of the matrix substantially. It gives better result than clustering, as it keeps most of the information of documents and terms, then compare the similarity among them. But it also has some disadvantages: 1. Since it is a linear model, it might not do well on datasets with non-linear dependencies. 2. LSA assumes a Gaussian distribution of the terms in the documents, which may not be true for all problems. 3. LSA involves SVD, which is computationally intensive and hard to update as new data comes up. 4. Lack of interpretable embeddings (we don't know what the topics are, and the components may be arbitrarily positive/negative) 5. Need for a really large set of documents and vocabulary to get accurate results 6. It provides less efficient representation.

c) Latent Dirichlet Allocation (LDA)

Both Latent Semantic Analysis (LSA) Latent Dirichlet Allocation (LDA) and techniques are frequently used for topic modelling. Latent Semantic Analysis uses the SVD as the core method to extract features. But LDA provides full generative probabilistic semantics for documents. Documents are modeled via a hidden Dirichlet random variable that specifies a probability distribution on a latent, low dimensional topic space. The distribution over words of an unseen document is a continuous mixture over document space and a discrete mixture over all possible topics.

LDA assumes that there are k underlying latent topics according to which documents are generated, and that each topic is represented as a multinomial distribution over V words in the vocabulary. A document of N words $w =< w_1, \dots w_N >$ is generated by the following process:

i. θ is sampled from a Dirichlet $(\alpha_1, \dots, \alpha_k)$ distribution.
ii. For each of N words, a topic $z_n \in \{1, \dots k\}$ is sampled from a Multinomial distribution $p(\ z_n = i|\theta) = \theta_i$
iii. Each word w_n is sampled conditioned on the $z_n th$ topic from the multinomial distribution $p(\ w|z_n)$
iv. The probability of a document is therefore the mixture:

$$p(w) = \int_{\theta}^{N} \left(\prod_{n=1}^{N} \sum_{z_n=1}^{k} p(w_n|z_n;\beta) p(\ z_n|\theta) \right) p(\theta;\alpha) d\theta$$

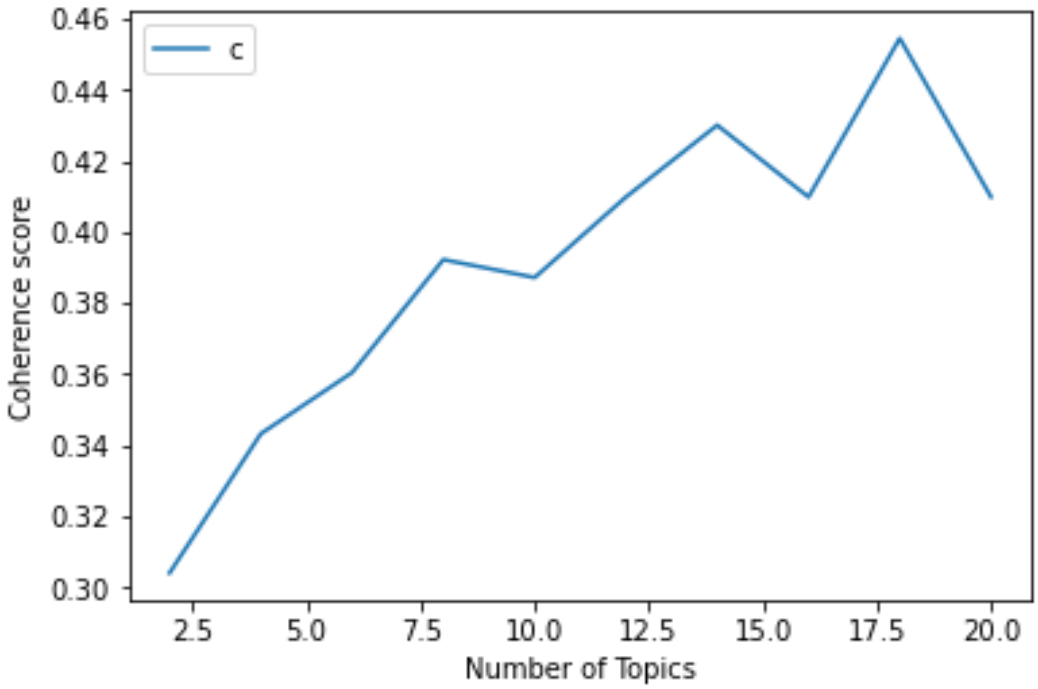
$p(\theta;\alpha)$ is Dirichlet, $p(\ z_n|\theta)$ is a multinomial parameterized by θ, and $p(w_n|z_n;\beta)$ is a multinomial over words.

When implementing, we decide the number of topics k. Each word that appears in the corpus is randomly assigned to one of the k topics. This assignment is technically not random, since it involves a Dirichlet distribution that employs a probability simplex instead of real numbers, which means that the numbers assigned across the k topics add up to 1.

Topic assignments for each word are updated in an iterative fashion by updating the prevalence of the word across the k topics, as well as the prevalence of the topics in the document. This stage of LDA also employs the Term Frequency-Inverse Document Frequency metric. Topic assignments are updated up to a user-specified threshold, or when iterations begin to have little impact on the probabilities assigned to each word in the corpus.

LDA also produces two types of output. First, one can identify the words that are most frequently associated with each of the k topics. Second, LDA produces the probability that each document within the corpus is associated with each of the k topics. In LDA, the Dirichlet is sampled for each document, and the multinomial topic node is sampled repeatedly instead of only once within the document. The Dirichlet is thus a component in the probability model rather than a prior distribution over model parameters. Thus, LDA is a mixture model.

In LDA, the number of topics also needs to be tuned. Here, coherence value is again used to quantify the quality of the number.



Entertainment	• (0.13043633, 'play'),(0.09887096, 'game'), (0.083998196, 'watch'), (0.039507613, 'music'),(0.03613897, 'video'), (0.027551861, 'show'), (0.02501758, 'character'), (0.023867736, 'movie'), (0.019139674, 'movies'), (0.016329713, 'enjoy'), (0.011040564, 'listen'), (0.010745591, 'anime'), (0.009219073, 'bore'), (0.009090188, 'songs'), (0.008504981, 'sport'),(0.008305516, 'favorite'), (0.007795873, 'countries'), (0.007147264, 'incel'),(0.007136531, 'team'), (0.006197795, 'football')
Career Advancement	• (0.061432365, 'ampx200b'), (0.02809161, 'country'), (0.022787167, 'english'), (0.02212272, 'society'), (0.016085401, 'write'), (0.0160803, 'work'), (0.01578344, 'language'), (0.015638316, 'learn'), (0.013595967, 'study'), (0.01242766, 'career'), (0.01173966, 'interview'), (0.011278861, 'apply'), (0.0096113775, 'degree'), (0.008082638, 'dream'), (0.007933818, 'company'), (0.0077988394, 'experience'), (0.0075569744, 'master'),(0.0075078076, 'field'),(0.0073687676, 'job'), (0.0066337907, 'program')
Financial Issues	• (0.076882966, 'money'), (0.02749725, 'pay'), (0.02128601, 'work'), (0.015678985, 'give'), (0.01543897, 'buy'), (0.0136931855, 'sell'), (0.013139542, 'save'), (0.012571944, 'bill'), (0.0124511, 'store'), (0.011864177, 'steal'), (0.011618169, 'card'), (0.011104878, 'make'), (0.010926958, 'rent'), (0.010156364, 'spend'), (0.010033218, 'need'), (0.008752241, 'afford'), (0.008646266, 'business'), (0.008196937, 'company'), (0.007701163, 'gift'), (0.0074024103, 'house')
Relationship	• (0.042320576, 'women'), (0.03690398, 'girls'), (0.034965344, 'girl'), (0.031342126, 'look'), (0.025152113, 'sexual'), (0.022683494, 'woman'), (0.020727552, 'guy'), (0.020293443, 'porn'), (0.018636493, 'date'), (0.016694536, 'body'), (0.015015568, 'female'), (0.0148961935, 'male'), (0.013671054, 'attract'), (0.012729264, 'attractive'), (0.012387184, 'find'), (0.009124181, 'sexually'), (0.008645362, 'like'), (0.008502891, 'straight'), (0.00850048, 'dick'), (0.008015395, 'horny')
Food	• (0.074582696, 'food'), (0.053916935, 'eat'), (0.033221904, 'dog'), (0.024712771, 'cook'), (0.02411637, 'animals'), (0.021448221, 'cat'), (0.019404797, 'animal'), (0.014311133, 'taste'), (0.011348009, 'pet'), (0.011315756, 'meat'), (0.011159348, 'pizza'), (0.010830144, 'americans'), (0.0107139675, 'chicken'), (0.010471676, 'massage'), (0.009896143, 'dinner'), (0.009892042, 'milk'), (0.009585459, 'hungry'), (0.009544841, 'meal'), (0.009489228, 'cheese'), (0.0093584, 'fish')
Social	• (0.062158052, 'people'), (0.019445084, 'like'), (0.014554828, 'make'), (0.014078619, 'post'), (0.0102630025, 'think'), (0.007412575, 'say'), (0.007261087, 'even'), (0.007099027, 'person'), (0.0065210164, 'hate'), (0.006307755, 'something'), (0.0062631024, 'many'), (0.00596274, 'comment'), (0.0058529484, 'theyre'), (0.005834711, 'social'), (0.0057832757, 'someone'), (0.0057210387, 'mean'), (0.005692118, 'white'), (0.005285377, 'black'), (0.005076864, 'believe'), (0.005058748, 'know')

Evaluation and Final Results:

The project is an unsupervised learning project, with no true labeled values to test if our clusters algorithm has good performance. So, there are mainly two methods are used to evaluate the model.

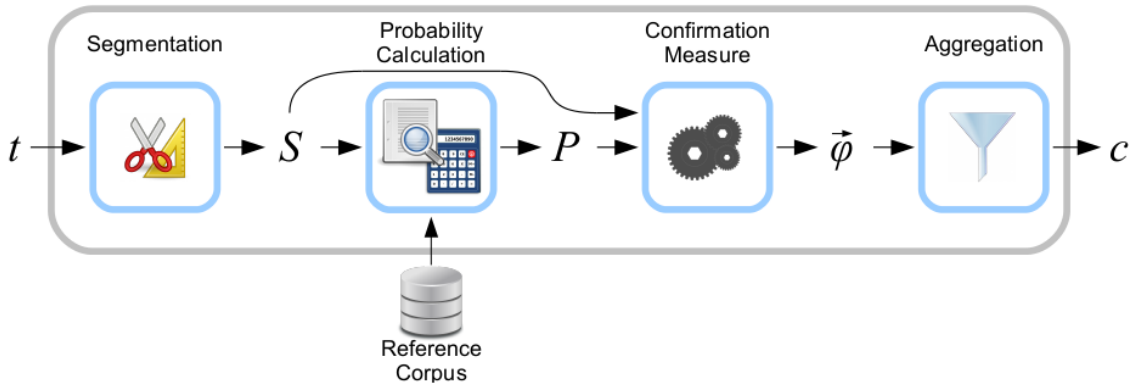
First, the silhouette coefficient evaluates the distance between clusters. If the clustering separates dissimilar documents apart and similar documents together, the value is close to 1, then it has performed well. The silhouette coefficient is calculated as:

$$s = \frac{b - a}{\max(a, b)}$$

a: The mean distance between a sample and all other points in the same cluster.
b: The mean distance between a sample and all other points in the next nearest cluster.
The score is bounded between -1 for incorrect clustering and +1 for highly dense clustering. Scores around zero indicate overlapping clusters. The score is higher when clusters are dense and well separated, which relates to a standard concept of a cluster. In the Kmeans clustering model, the value is very close to 0. It indicates that there are overlapping topics.

Second, it is coherence value. The high value of topic coherence score model will be considered as a good topic model. There are four parts in calculating coherence value: Segmentation, Probability Estimation, Confirmation measure and Aggregation.

i. Segmentation: Where topics are partitioned into several parts assuming that the probabilities of topics in each part is different.
ii. Probability Estimation: Where the probability of topics in each part is measured.
iii. Confirmation Measure: Where the probability of topics in each part is measured and a number is assigned to each part wrt it's probability.
iv. Aggregation: The value where these probabilities are combined in a certain way (say arithmetic mean) to come up with one number.



t: Topics coming in from the topic model
S: Segmented topics
P: Calculated probabilities
Phi vector: A vector of the "confirmed measures" coming out from the confirmation module
c: The final coherence value

In LSA and LDA, I tuned the parameter by using the coherence value. We can also compare these two models. From the result of coherence values, the LDA model words better than the LSA and Kmeans model. The interpretation is also easier. By looking at the topics, LDA has fewer overlapping terms in different topics.

Model	*No. of Topics*	*Evaluation Result*	*Advantages*	*Disadvantages*
Kmeans	20	-0.00032	1. Documents with same words are clustered together 2. Implementation is easy	1. Overlapping documents and terms 2. Interpretability is bad. 3. No hidden topics extracted
LSA	8	0.389	1. Dimension reduction	1. Solution is linear.

			2. Hidden topics are extracted	2. Interpretations of the scalar values are hard.
LDA	18	0.454	1. Mixture model: documents and terms can appear in different topics. 2. LDA works efficiently, online learning is available.	1. There are many parameters to tune to decide the best model. 2. When sample size is too small, LDA does not necessarily perform well.

Challenges and Limitations:

1. Memory issues: The biggest challenge for this project is memory problem as there are 1 million of threads. The csv file alone takes more than 1 GB of memory. The implementation is very difficult as it is often a failing result. However, I managed to get the result of the project by doing the following:
    a. Clean the data: Remove uninformative and excessive information. In the preprocessing, I removed the deleted and removed answers.
    b. Choose a proper package: Getting tf-idf matrix could be very easy for small-sized data in Sklearn, however the function stores its dictionary along making a matrix. So, it is impossible to run whole data using the function. An alternative is to use Gensim to build the matrix. Gensim performs fast truncated SVD. During SVD decomposition, the result can be updated with new observations at any time, for an online, incremental, memory-efficient training. So even a large data still fits, as only constant memory is needed.
    c. Batching: In the course, we learned bagging and batching are a good way to combine learners and get an average result of models. So, in Kmeans clustering, I choose to use Minibatch, and in LSA and LDA models, I also use chunk size to help reduce the memory used and improve the speed of running.
    d. Running in chunks: Instead of using a large chunk in the notebook, I use small chunks.
2. Parameter tunning: In this project, I only tuned the number of topics in three models. But these models can have other parameters tuned and the results could be very different. For example, in LDA model, alpha 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. Beta parameter 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. I set these parameters to “auto” but it could be also tuned. Other parameters might also affect the model results such as chunk size, or batch size, iterations, evaluations, random state etc. When evaluate the results with coherence models, there are also different evaluation metrics can be implemented, I use “c_v” method as it is more accurate than “u_mass”.
3. Model selection: Several associated techniques can also be tested, including Dynamic Topic Models, Correlated Topic Models, Hierarchical Topic Models, and Structural Topic Modeling.
4. Inferring topics from key words: We can decide the optimal number of topics, but the interpretation of results is arbitrary. With only key words, we humans need to infer the topics behind the keywords, but interpretation can be different. The term “topic” is somewhat ambiguous, and it is perhaps true that topic models will not produce highly nuanced classification of texts. Another drawback is that the modelling techniques should not be the only way to rely on. Topic models provide a good tool for reading and summarize documents. But it should not be the only way for us to decide the number of topics. As the preprocessed data, and other elements can also influence the result. That’s also why preprocessing is very important for modelling. For example, the word “theyre” should be cleaned but the preprocessing cannot rule out all the mistyped words or “lazy” writing. So, when examining the results, both the quantitative and qualitative techniques should be implemented.

Reference:

public.pdf (aksw.org)
nips2013tm_submission_7.pdf (cornell.edu)
nips01-lda.pdf (stanford.edu)

Latent Semantic Indexing: An overview (msu.edu)
dp1.LSAintro.pdf (colorado.edu)
4.pdf (stanford.edu)
Latent Semantic Analysis and its Uses in Natural Language Processing (analyticsvidhya.com)
News documents clustering using python (latent semantic analysis) | by Abhijeet Khangarot | Kuzok | Medium
Latent Semantic Analysis — Deduce the hidden topic from the document | by Sanket Doshi | Towards Data Science
Python LSI/LSA (Latent Semantic Indexing/Analysis) - DataCamp
Stemming and Lemmatization in Python - DataCamp
Preprocess Text in Python --- A Cleaner and Faster Approach | Think.Data.Science (thinkdatascience.com)
Speeding up text pre-processing using Dask | by Shikhar Chauhan | MindOrks | Medium
Introduction to Latent Dirichlet Allocation (echen.me)
Natural Language Processing and Topic Modeling on User Review Dataset | by Yanhan Si | Level Up Coding (gitconnected.com)
Online Learning for Latent Dirichlet Allocation (neurips.cc)
What is Topic Coherence? | RARE Technologies (rare-technologies.com)
Topic Modeling and Latent Dirichlet Allocation (LDA) using Gensim (analyticsvidhya.com)
Gensim Topic Modeling - A Guide to Building Best LDA models (machinelearningplus.com)
Analyzing Amazon TV reviews with Latent Dirichlet Allocation | by George Boben | Analytics Vidhya | Medium
Multicore LDA in Python: from over-night to over-lunch | RARE Technologies (rare-technologies.com)
Topic Modeling in Python: Latent Dirichlet Allocation (LDA) | by Shashank Kapadia | Towards Data Science
towardsdatascience.com
LSA / PLSA / LDA (ppasupat.github.io)
The Ultimate Guide to Clustering Algorithms and Topic Modeling | by Zijing Zhu | Towards Data Science
K-Means Clustering. Making Sense of Text Data using… | by Daniel Foley | Towards Data Science
Latent Semantic Analysis: intuition, math, implementation | by Ioana | Towards Data Science