Web Browsing Behavior Analysis

Genghua Chen

Table of Content

Introduction **2**

Data collection and Understanding **3**

Data Preprocessing **9**

Experiments and Analysis **12**

Embedding 12

Clustering 25

K-Means 26

Experimental Analysis and Evaluation For K-Means 26

Gaussian Mixture Model 28

Experimental Analysis and Evaluation For GMM 28

HDBSCAN 30

Experimental Analysis and Evaluation For HDBSCAN 30

Clustering Results 31

Results **34**

Conclusion/ Discussion **37**

Introduction

The Internet plays a huge role in today's society. We go online almost every day to find the things we need to help us get our work done. The Internet is even more important during the pandemic, due to the government policy and everyone staying home, and people taking classes from home via the Internet. Then student behavior is a concern for school and parents. This becomes more challenging, and then the activity becomes more concerned because teachers and parents don't know what students are actually doing while using the browser. The student could play games while studying. If the teacher or parent doesn't keep their eyes on them all the time, the students can totally switch to a gaming website. Understanding students' online behavior is crucial for teachers and parents alike, and this program will provide a better understanding of what students are thinking and keep them away from bad websites, exactly what teachers and parents want to do. One solution is to block websites that teachers and parents disapprove of so that students cannot visit them. We try to know what websites the student is browsing through in this project.

The scope of this project is limited to online browsing behavior and does not include the use of other applications on the computer, as such data are not readily available. The main goal of this project is to develop a system for analysis of browsing behavior patterns and use interactive visualization techniques to promote reflections to parents and teachers. In the first step, we will roughly brow through the titles, and then clean the titles, such as removing symbols or non-English words. Next, we embed the title and/or abstract of a web page into a high-dimensional vector space, using models such as the Glove pre-trained model, to obtain the vector representation of the titles. And then we experimented with several dimension reduction techniques, such as linear embedding and non-linear embedding methods so that we can visualize them. Next, we apply the clustering technique to cluster these embedded vectors into groups. Web pages containing similar semantics should be clustered into the same group by using several popular clustering algorithms. Lastly, we visualized patterns and knowledge discovered based on the groups retrieved from clustering by using the word cloud.

Data collection and Understanding

The dataset was collected from a middle school script through students browsing history. The dataset has 3,895,798 rows and 10 columns:

- host: Domain of a website
- url: URL of a website
- title: Title of a website
- type: Either activities or violations. A violation occurs when the website gets blocked as students browse.
- category: The category of a website, generated by domain.
- ccCates: The ID of the category

- timestamp: The time that a student starts browsing a website
- elapsedTime: Duration on a website
- user: ID of the students
- isOffCampus: Whether students are browsing on campus or not.

Before everything starts, we need to understand the data better.

- Distribution of categorical columns
    - The purpose is to understand how the distribution represents categorical columns.
- Daily student browsing behavior
    - The purpose is to see what a student's browsing behavior is like and have a sense of the different categories.

We did a user behavior plot about the data, such as looking up the distribution of category columns. We also created a plot to see the daily activities of a student. The purpose is to understand students' browsing behavior in a day and the distribution of the categorical columns.

The x-axis and y-axis are the time from 0:00 to 23:59. The many horizontal lines represent the duration of a web page. Square and triangles represent the time to open and close the web page respectively, while colors represent different categories. Finally, the red star represents the violation, meaning that this student opened something he/she shouldn't open. The purpose of this visualization is to make information on the plot accessible and clear. However, the daily behavior plot is not enough since it only tells us one perspective of the student's behaviors. We also need a zoom-in plot and focus on a specific period of time to see what the student is doing. Figure 1 is one example, it shows a student's daily browsing behavior in a day via a 2 dimensions plot. Figure 2 is the specific time of that student's activities between this time.

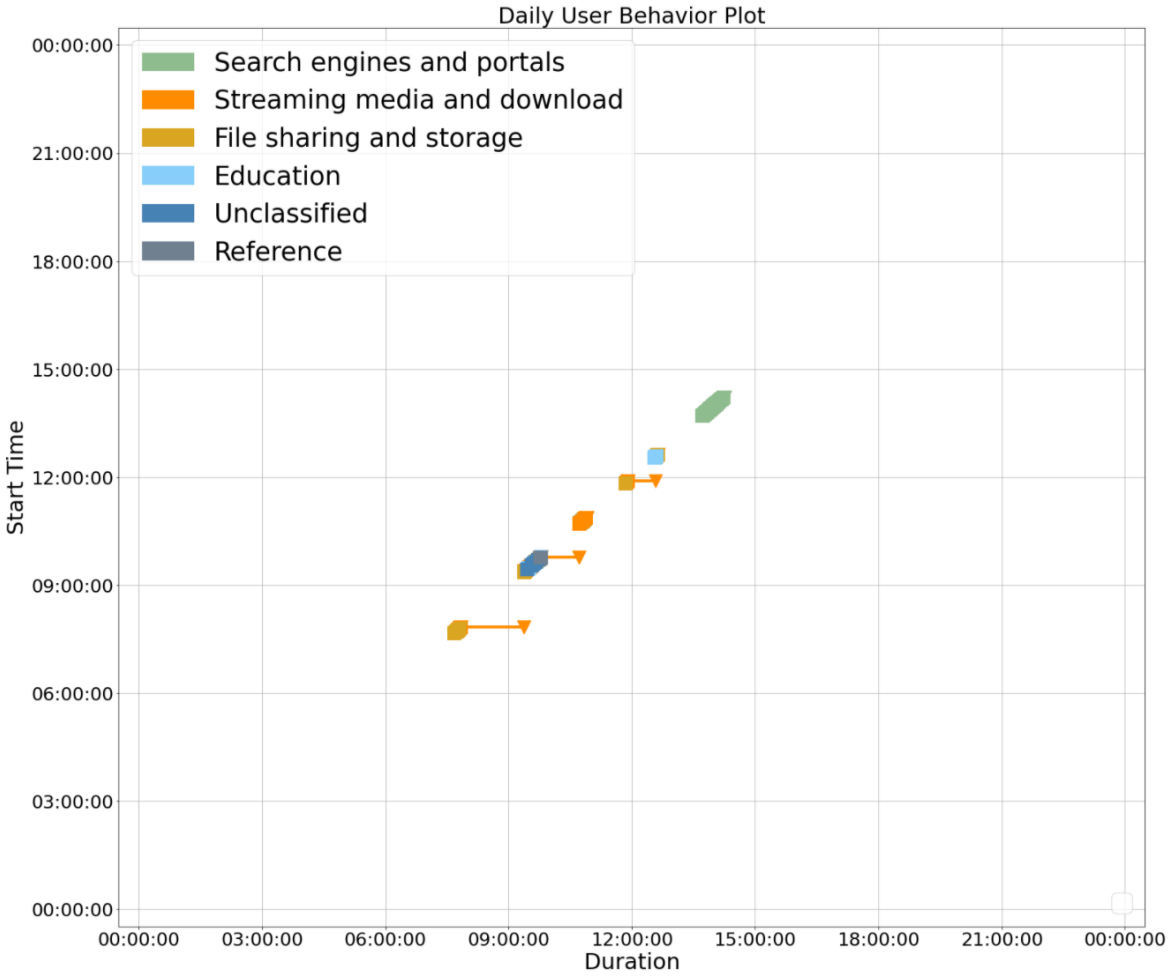

Figure 1: Daily User Behavior Plot in 24 hours

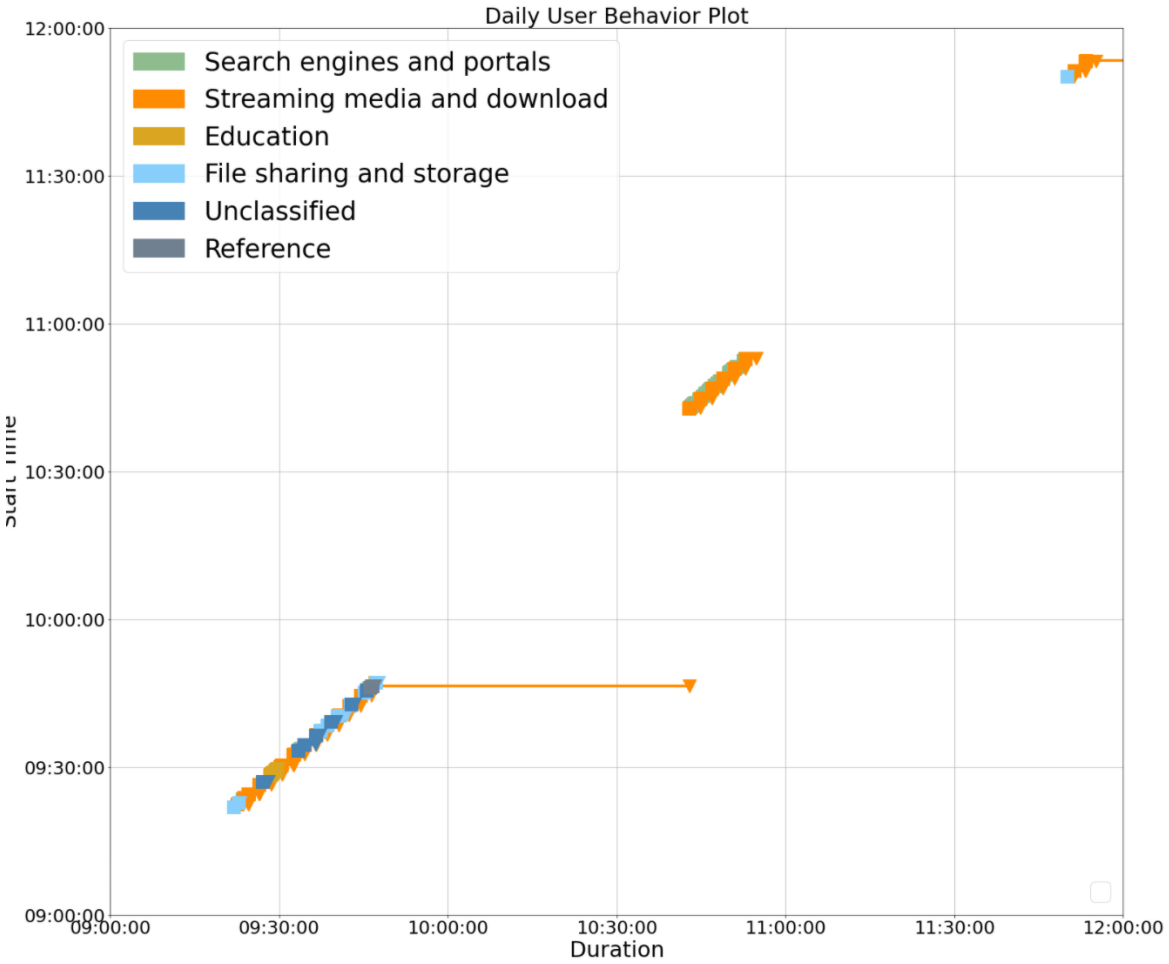

Figure 2: Zoom in Daily User Behavior Plot between

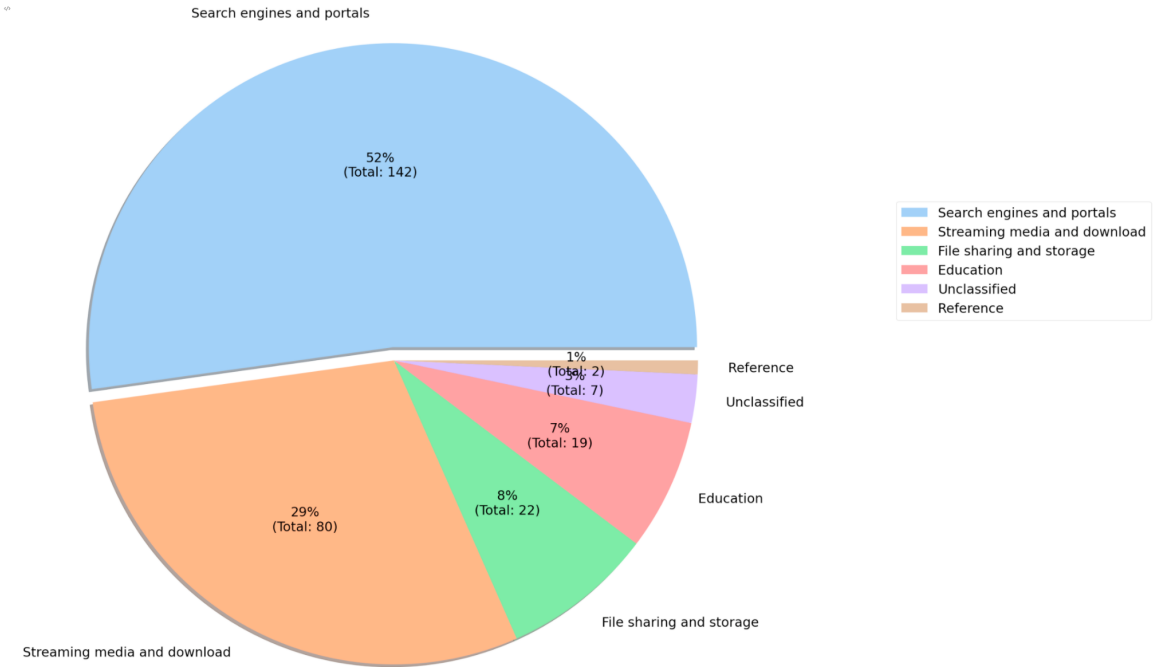

Figure 3: Proportion of Specific Daily User Behavior in Each Category

In our case, we care about the distribution in the category columns, which gives us a sense and expectation of the clustering result. There are 82 unique categories in total, but we only show the top 30 categories because we can see in Figure 4 that most of the data are from categories: ‘Education’, ‘Search engines and portals, ‘File sharing and storage, and ‘Streaming media and download’.

The hard labels are generated by the machine by domain. For example, the “classroom.google.com” domain is the Education category. And we can see that the seventh category is the ‘Unclassified’ category, which means the web crawler cannot distinguish what the labels are for those titles. So we realized that not all the hard labels are correct, then that is why we need to group them by using the clustering method, which is unsupervised learning, and the original labels only for reference.

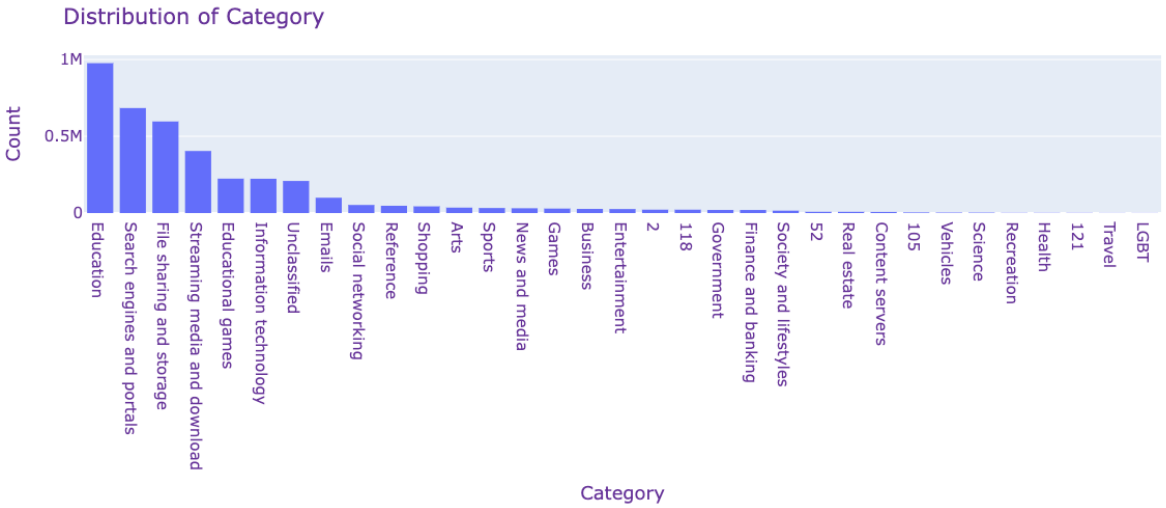

Figure 4: Distribution of Category Column

Now, we know that more than 50 percent of the data are ‘Education’, ‘Search engines and portals, ‘File sharing and storage, and ‘Streaming media and download’ categories. We are interested in all the categories however due to data availability or complexity, we will focus on the top categories which have more data points for us to analyze. Figure 5 to Figure 8 shows the distribution of the domain in the top 4 categories. There are only a few significant domains. Such as Google docs, YouTube, Spotify, and Google classroom. Also, there is a domain we notice that should be in the “File sharing and storage” category is “memedriod.com”, after visiting that website, we found that the website is similar to Tiktok and consider it as a “Streaming media and download” category. This demonstrates that applying the clustering method is necessary.

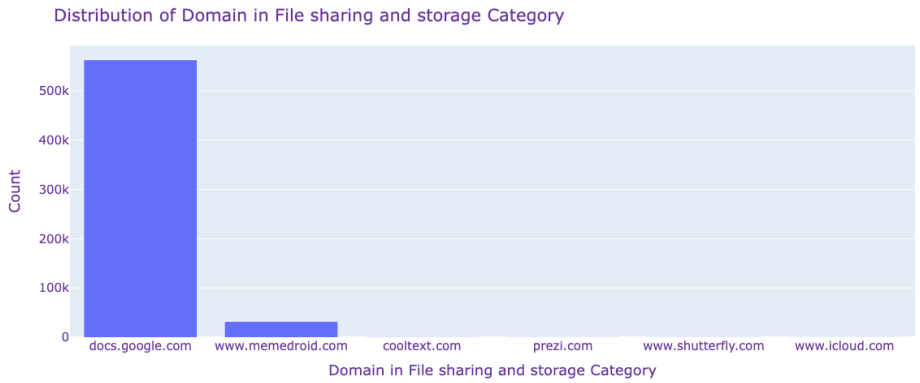

Figure 5: Domain in File Sharing and Storage

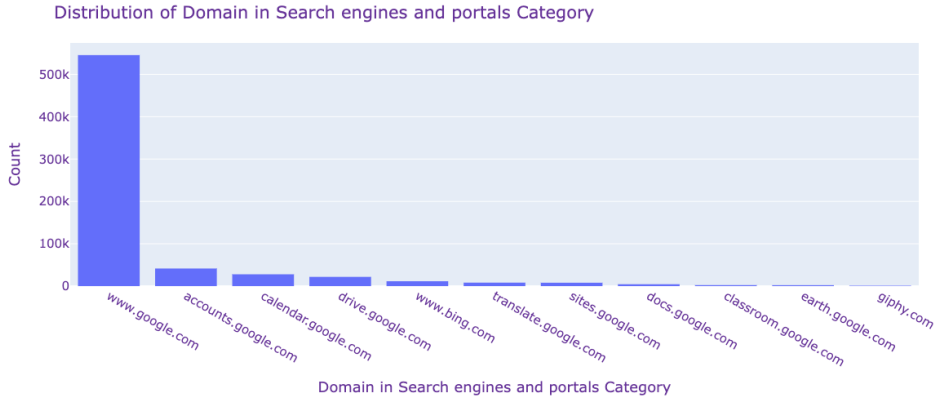

Figure 6: Domain in Search Engines and Portals

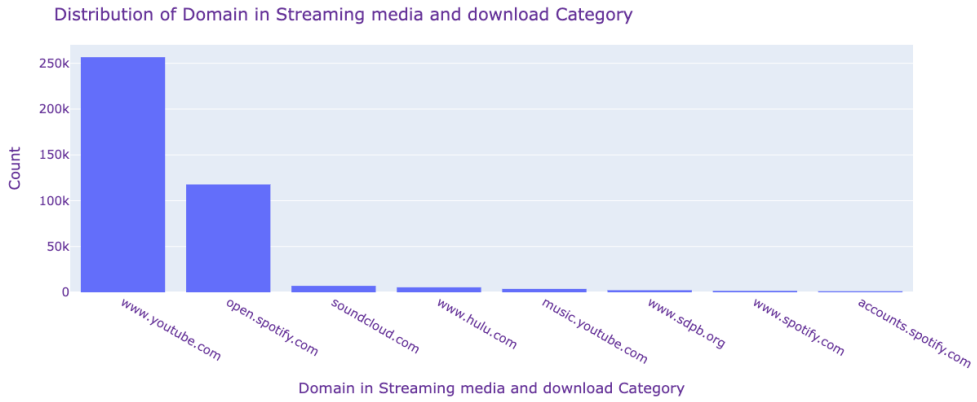

Figure 7: Domain in Streaming Media and Download

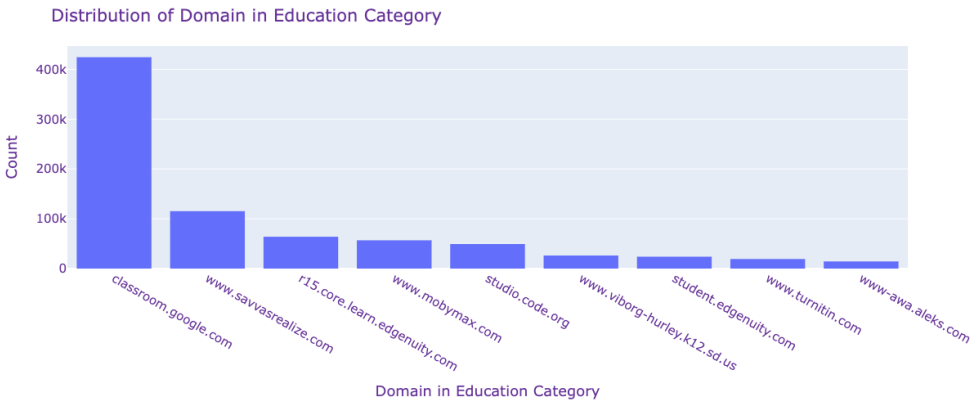

Figure 8: Domain in Education

Data Preprocessing

Next step is the main part of the project. We have to figure out a way to separate school-related websites and non-school-related websites by using clustering methods to visualize. The rough idea is to use an existing pre-train word embedding model and then get the vector representation for each title. And then apply embedding and clustering methods for each title.

There's a lot of punctuation, Japanese, serial numbers, etc… Before we convert it into a vector representation, we have to clean the title first and try to get as much information from the original titles.

Firstly, we have to parse all the titles and remove the titles containing ‘Google Search’ because those titles are very difficult to see what category it should be in. The method we used is converting all the letters to lowercase because the one we used has no uppercase letter in the Glove dictionary since there is no different meaning between lowercase and uppercase in our case. And then replace punctuation with space because we need to get as much information as we can. Even though some useless words may appear, they may not be in the Glove dictionary. However, we used a Python package called stopwords from *nltc,* and we made it as a text file to store the words which we don’t think are meaningful and delete them during the title cleaning process. And then we remove all the Non-English words, such as Chinese, Japanese, Russian, etc… Also, we remove all the numbers and the words containing numbers because after looking into the titles, we found that the numbers do not mean anything.

Let's look at some examples of how the title parsing works in Table 1. We replaced the U.S. with american because we used no uppercase Glove Pre-train model, which caused the word “U.S.” to “us” or “u s” and just change the meaning of the whole world,

and this title shows many times, it may affect the result. Hence, the best way is to replace it with american. Additionally, there is a course number “SS3311”, it might not be in the Glove Pre-train model, but is a good example to show that without the number it will be better if we remove it, as well as the year “2014”. There are some titles that only have non-English words such as the second example, the only word we can extract is “Youtube”, which is not enough information to represent this title. In future work, maybe we can translate those titles to English and extract useful information. But we just leave it for now. If there is not even an English word in a title, we just skip it. The third, fourth, and fifth examples are successful in extracting information after parsing.

No.	Before Title Cleaning	After Title Cleaning
1	U.S. History II 2014 - SS3311 B-IC - Edgenuity.com	american history ii edgenuity
2	ゴクウブラックはトランクスを過去にさかのぼって追跡します、ゴクウブラックが超サイヤ人2悟空と戦って怪我をする - YouTube	youtube
3	Technical Writing/Strategic Reading	technical writing strategic reading
4	20-21 Wars & Conflicts/Current Events	wars conflicts current events
5	Outlook help & learning - Microsoft Support	outlook help learning microsoft support

Table 1: Title Cleaning Examples

And then we converted each title into vector representation by using the Glove pre-train dictionary and see the formula of method (1)

$$m = \frac{v_1 + v_2 + \dots + v_n}{n} \quad (1)$$

n: $represents\ the\ total\ number\ of\ the\ words\ in\ this\ title$

v: $represents\ the\ vector\ representation\ for\ a\ word$

However, it can lead to inaccurate vector representation of a title if a title is not clean enough, but it is also a standard way of doing it.

The reason of used the Glove pre-train model is that it has shown the state of the art performance in any of the models. In our cases, our focus is not to find the best vector representation, but rather than finding a suitable way to embedding and clustering.

We experimented on only using titles and tried to cluster those similar titles together, but we realized that the result is not really representative, not convince people, such as scratch, we know it is a website for students to learn coding, but there are also many games can play. So we try to see if there is some useful information in the URL, which we can use and make the clusters better. We parsed the URL and then added some

weights to it and added after titles, and see if the clustering is going to be better or not. This formula is based on formula (1). See the new formula (2).

$$N = \frac{\alpha m_1 + \beta m_2}{\alpha + \beta} \quad (2)$$

N: $Weighted\ average\ equation$

v: $The\ vector\ representation\ for\ a\ word\ in\ title$

m: $The\ vector\ representation\ for\ a\ word\ in\ url$

Based on formula (2), we can simplify to formula (3).

$$n = \frac{m_1 + \alpha m_2}{1 + \alpha} \quad (3)$$

The method of parsing URLs and titles is different, the purpose is to get as much information as we can. During parsing the URL, we get all the domains out and then replace slash, dash, and underscore with space. The rest of the other symbols are just ignored because there will be too many noisy words.

It's challenging to parse the URL, although there is a lot of useful information, on the other hand, there is a lot of useless information which will affect the result. The first example below is a good example to prove this parse works. There are many keywords that are important, such as science, elements, life or earth. These keywords may be part of the titles, but we have more useful information to represent a link. But the second example is a bad example of our parse method. Thus, we decided to use a stopword from nltk.corpus to omit those useless words, such as the, looking, adcb, a or on, etc… and make the URL more representative.

Before Url Cleaning Examples	After Url Cleaning Examples	Word in Glove Dictionary
https://www.newyorker.com/science/elements/looking-for-life-on-a-flat-earth	newyorker science elements life flat earth	science elements life flat earth
https://student.edgenuity.com/?returnOrder=3646e94b-b7ec-e311-adcb-b265d2b44129	student	student
https://www.infinitecampus.com/audience/parents-students/login-search	infinitecampus audience parents students login	audience parents students login
https://www.nfhsnetwork.com/events/parker-high-school-parker-sd/gam3944e90f1e	nfhsnetwork events parker high school	events parker high school
https://www.agdaily.com/vi	agdaily video musician	video musician captures

deo/musician-captures-ame rican-farmer-video/	captures american farmer video	american farmer

Table 2: Url Cleaning Examples

Since the dataset has 3 million rows, it would be hard to visualize if we used all the data and the computation cost of dimension reduction and clustering will be very high. We decided to sample 10,000 data and it will be easier to visualize and take less time for computation. We experimented with a couple of sampling methods, such as simple random samples and sample by portion for each category. The simple random sample is the easiest one, but it would not make sense, because we know that the top four categories for more than 50%, which means it has a higher probability to sample 10,000 in these four categories. But if we sample by portion from the original data, it won't have this problem. See Figure 9.

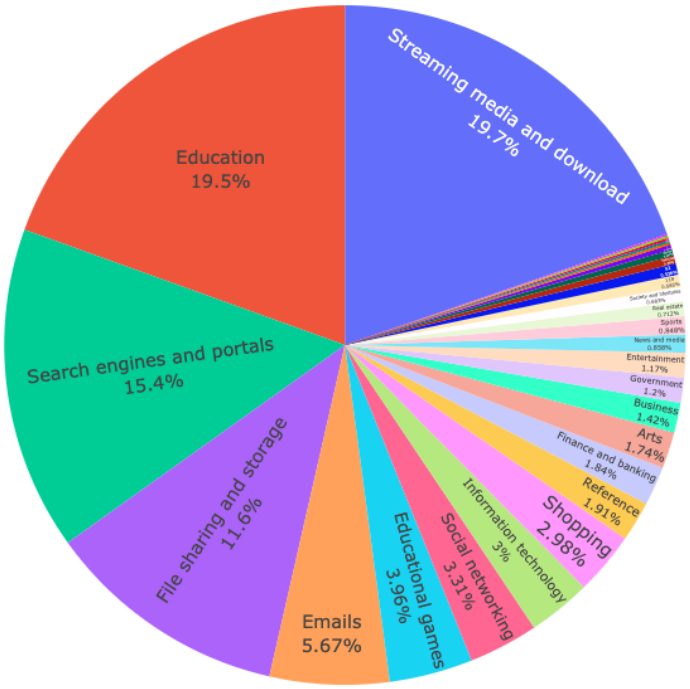

Figure 9: Percentage of every category

Experiments and Analysis

Embedding

By now, we have the vector representation for each title ready. The next step is going to visualize them in the embedding space since we cannot visualize them in high dimensions. We can transform the data from a high dimensional space into a low dimensional space so that the low dimensional representation retains some meaningful

properties of the original data. In our case, the vector representation should be embedded from 50 dimensions into either 2 or 3 dimensions.

We tested 4 dimension reduction methods for embedding methods, including two linear and two non-linear embedding methods, which are PCA, Multidimensional Scaling, T-SNE, and Locally Linear Embedding. The purpose is to find out which dimension reduction method extracts at most information, and then we can move to the next step. However, the dimension reduction would lose information for sure, the goal is to keep as much as we can and prepare for visualization. Also, we expected the clusters as detailed as possible. For example, the embedding method is able to distinguish ‘Google Docs’, ‘Goole PowerPoint’, ‘Goole Sheet’, and so on.

First, Principal Components Analysis is the most well-known unsupervised dimensionality reduction technique that constructs relevant features and is achieved by linearly transforming correlated variables into a smaller number of uncorrelated variables. In other words, PCA is a feature extraction technique. It combines the variables, and then it drops the least important variables while still retaining the valuable parts of the variables.

PCA is required to normalize before applying PCA, so we standardize data and then apply PCA. After reducing the dimension from 50 to 2 by using PCA, the 2D plot shows below. The variance ratio of PCA is 31%, which means that there is 31% of the information preserved in 2D, which is a significant information loss, but 31% might be the most important part. When we look at the plot, there is a huge area of points stuck together and the title in that area are similar, but some of the clusters are distinguished in the top right corner.

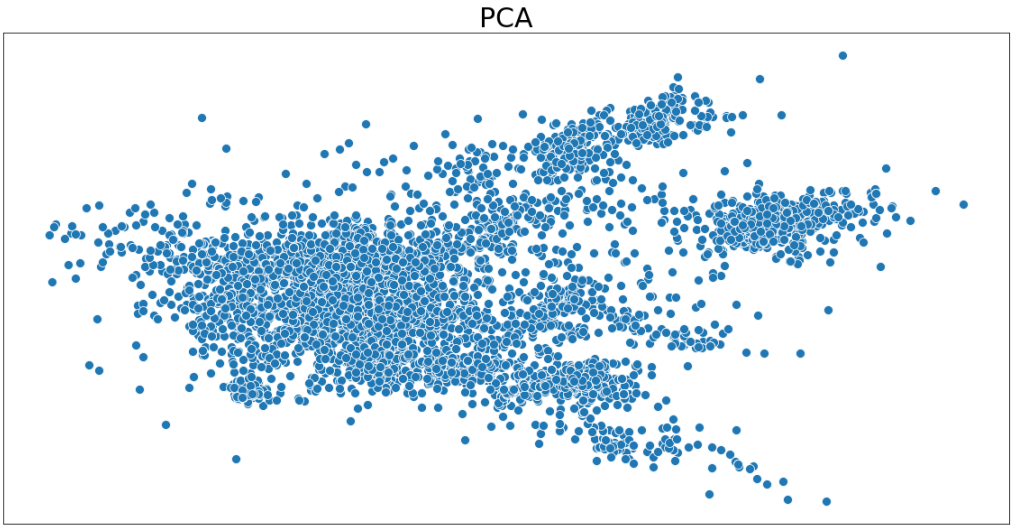

Figure 10: Visualization of 2D word Embedding using PCA

However, we can apply the clustering method based on PCA dimension reduction. But it has been challenging for any clustering method to distinguish all the categories for

sure. It might separate all the education correctly but it would include some of the other titles which should not belong in the education cluster, then the error will be huge.(See Figure 10) In other words, even humans cannot really separate it by drawing circles, how can a computer do it?

We wonder if PCA really performs in the correct way. The way to verify it is to take 10 sets of 10 points out randomly to visualize it, which 5 points are similar to each other but uncorrelated with the other 5 points. See Table 1. Spotify and Google Docs shouldn't be grouped together, because they are uncorrelated by intuition. Google Docs and Google sheets should be close to each other. 6th Grade English and Google Docs should be closer than Spotify. And the linear embedding method should have exactly the same x and y if the titles are exactly the same.

	x	y	title	category
81	2.685862	5.323291	Ghost in Town · Powfu	Streaming media and download
18	6.074792	3.209586	goodbye - YouTube	Streaming media and download
603084	4.234633	2.644030	Morgan Wallen - This Bar (Lyric Video)	2
6274	1.455763	-2.357532	All grade list - Google Sheets	File sharing and storage
2116	2.178956	-2.618137	Untitled document - Google Docs	File sharing and storage
2144	2.178956	-2.618137	Untitled document - Google Docs	File sharing and storage
1611	-2.039361	-0.966276	6th Grade English 6th	Education
6031	-1.318635	-1.517585	Classwork for 20-21 5th Technology	Education
0	-4.400167	-0.048615	U.S. History II 2014 - SS3311 B-IC - Edgenuity...	Education
7888	-5.721419	1.732638	Edgenuity - Student Learning Experience	Education

Table 3: Experiment of 10 titles for verifying the Embedding

For display purposes we only show the four out of ten verifications. In Figure 11 to Figure 14 shows that PCA works well on this sample because we can see that the same clusters are close to each other and if draw a few circles we can separate them out. Also, there is a gap between 'education' and 'streaming media and download', which is what we expected.

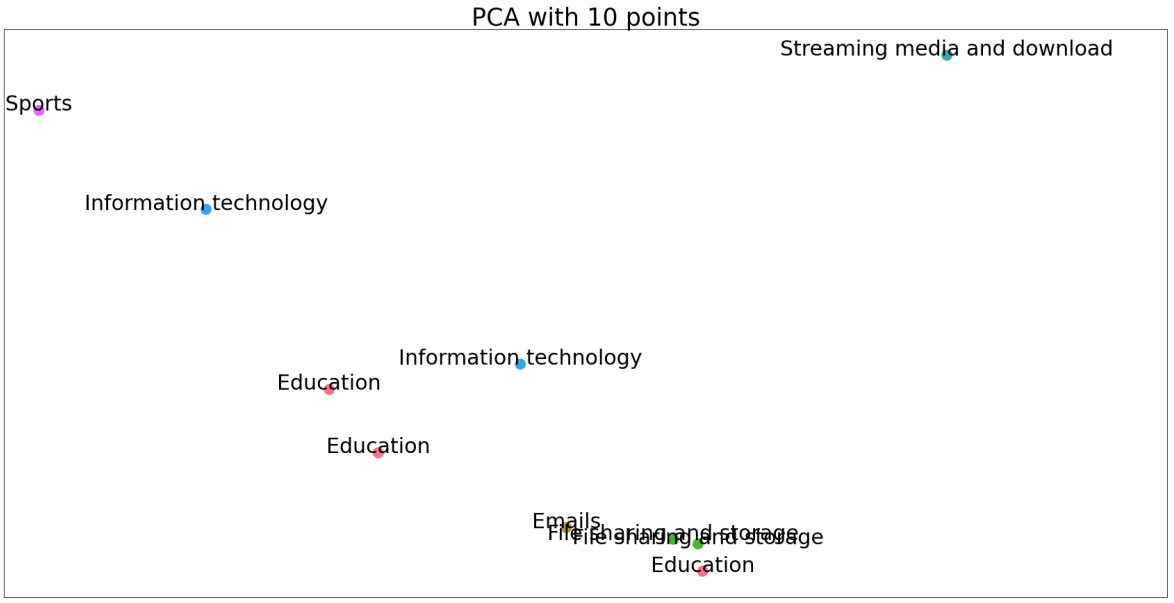

Figure 11: Visualization of 2D Word Embedding using PCA with 10 Points 1

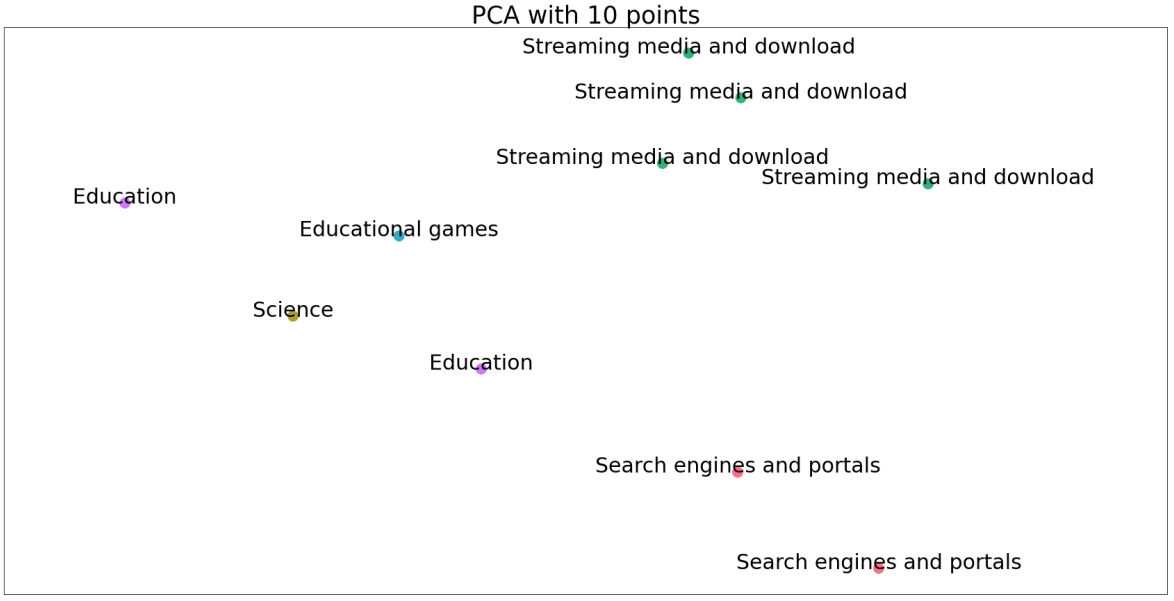

Figure 12: Visualization of 2D Word Embedding using PCA with 10 Points 2

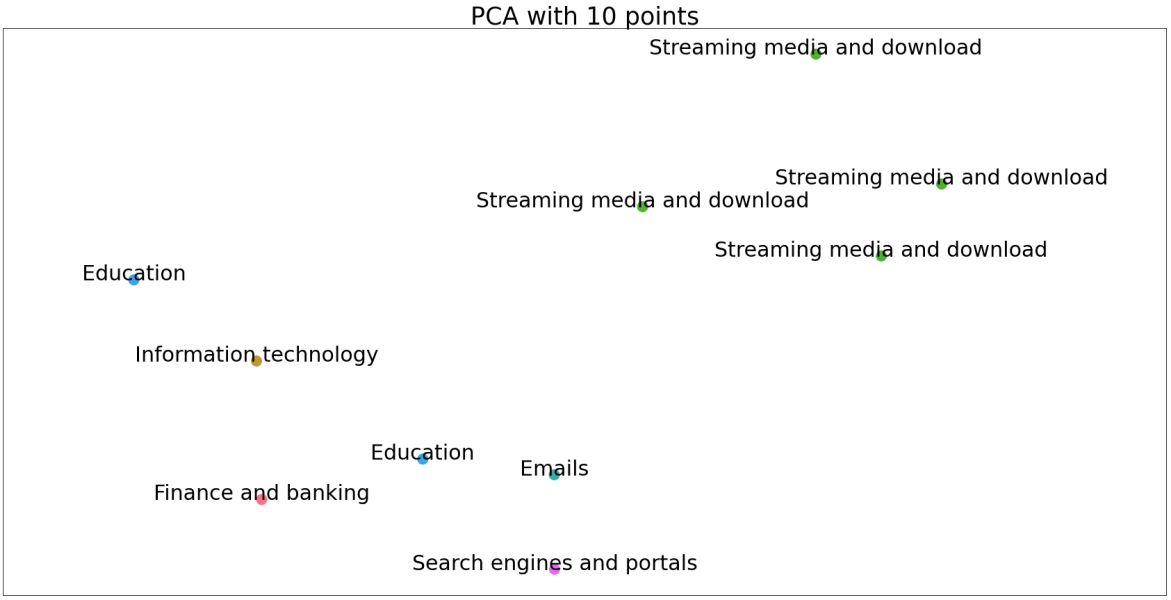

Figure 13: Visualization of 2D Word Embedding using PCA with 10 Points 3

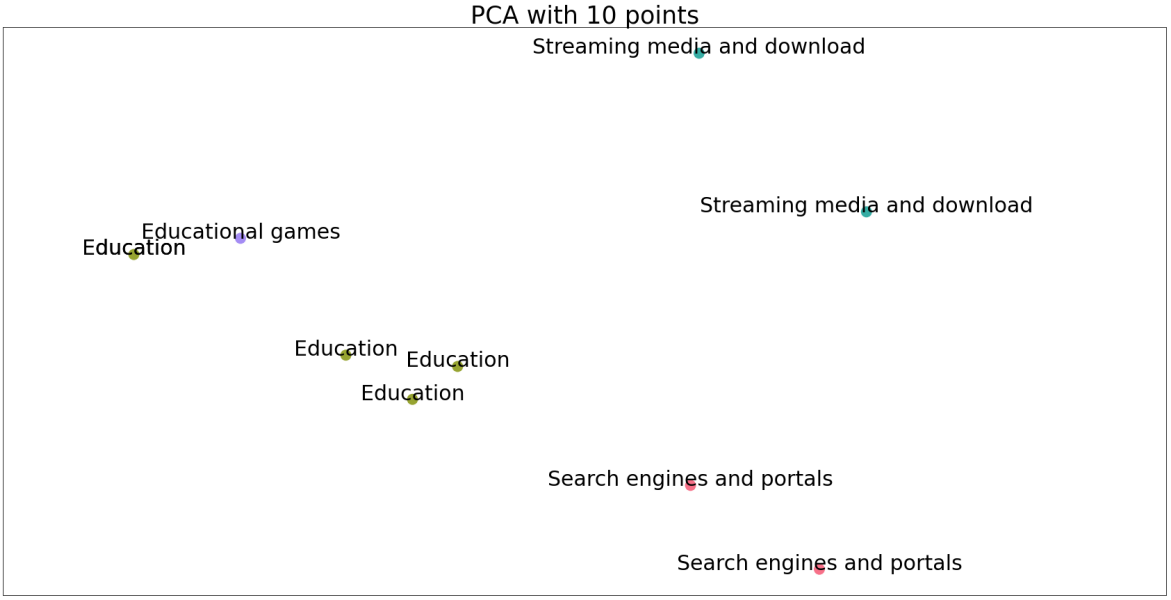

Figure 14: Visualization of 2D Word Embedding using PCA with 10 Points 4

Aside from PCA, another dimensionality reduction technique that is commonly used is Multidimensional Scaling (MDS), it is a linear method for dimensionality reduction, and it tries to preserve a measure of similarity between pairs of data points. MDS is very similar to PCA, classic Torgerson's metric MDS is actually done by transforming distances into similarities and performing PCA on those. So, PCA might be called the algorithm of the simplest MDS. We are using classical MDS, so the only hyper-parameter we can change is n_components.

In Figure 15, we can see the plot is very dense and like a ball, but it was able to distinguish some categories on the right part. Even so, we cannot just be saying that MDS is best than PCA or worst, so we need to verify it and compare it with PCA.

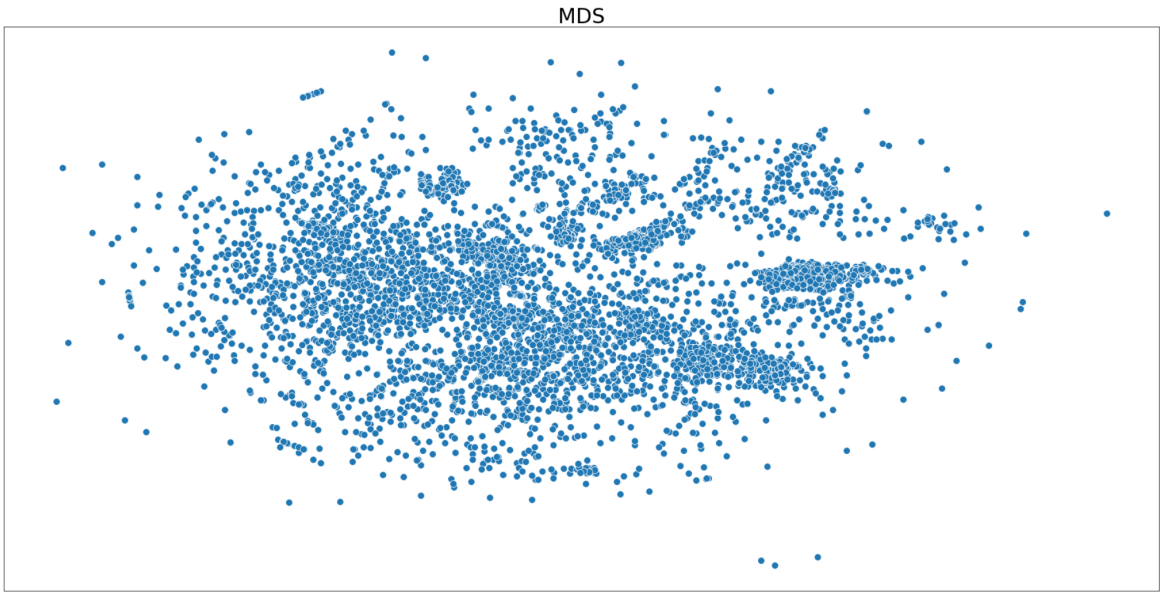

Figure 15: Visualization of 2D Word Embedding using MDS

The verification of MDS can distinguish between each category. We can see ‘Education’ on the top left corner and the ‘Streaming media and download’ on the bottom right corner. Nonetheless, it still can differentiate the different categories. (See Figure 15 to Figure 18)

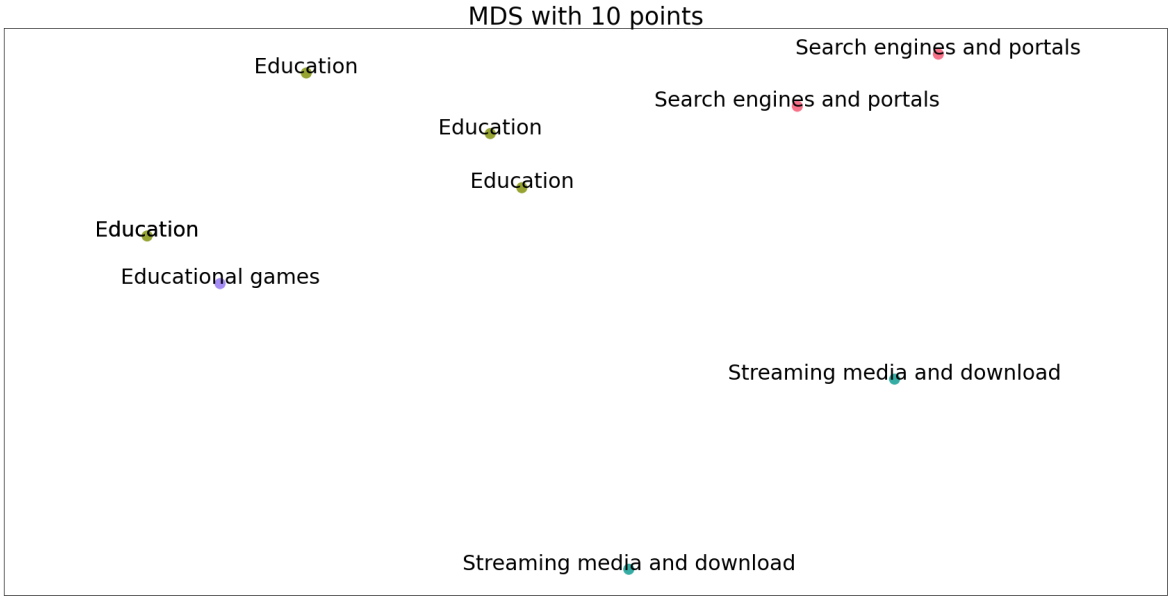

Figure 16: Visualization of 2D Word Embedding using MDS with 10 Points 1

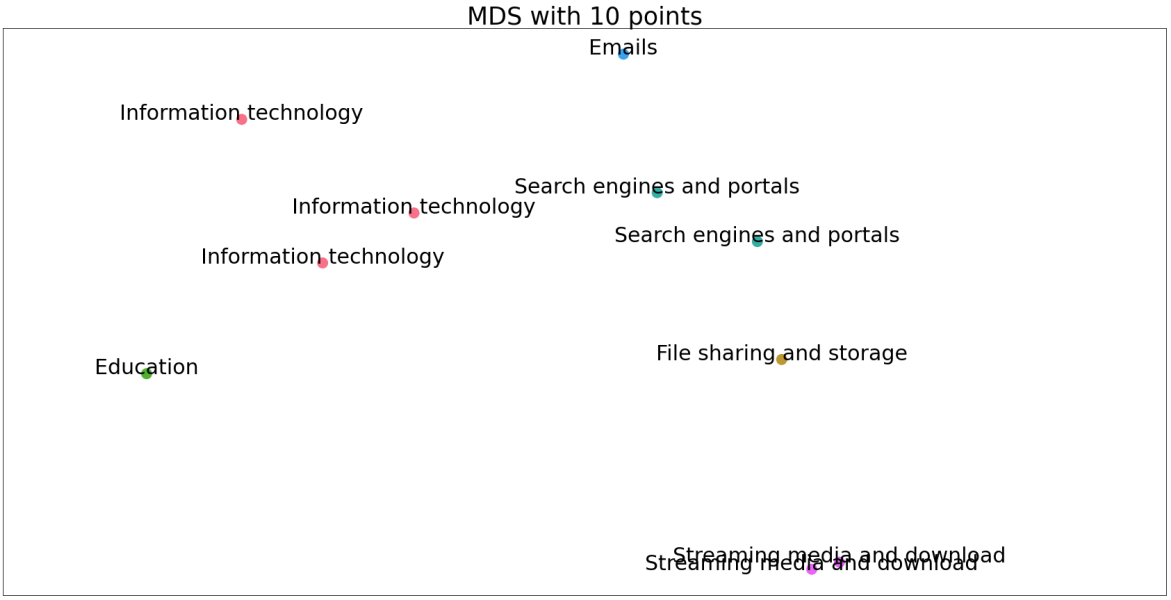

Figure 17: Visualization of 2D Word Embedding using MDS with 10 Points 2

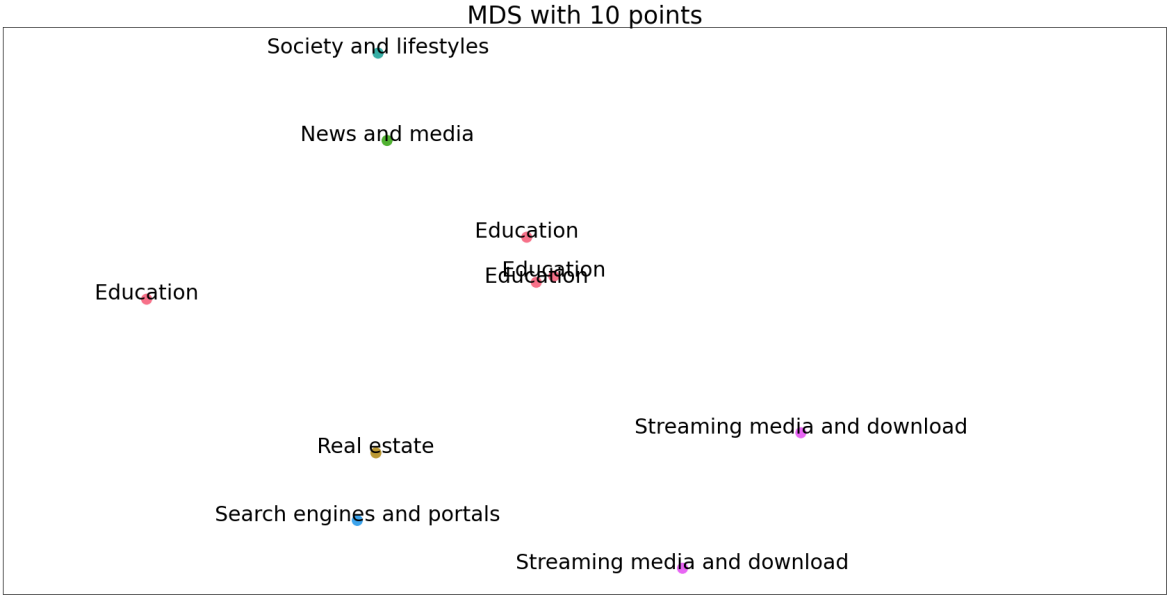

Figure 18: Visualization of 2D Word Embedding using MDS with 10 Points 3

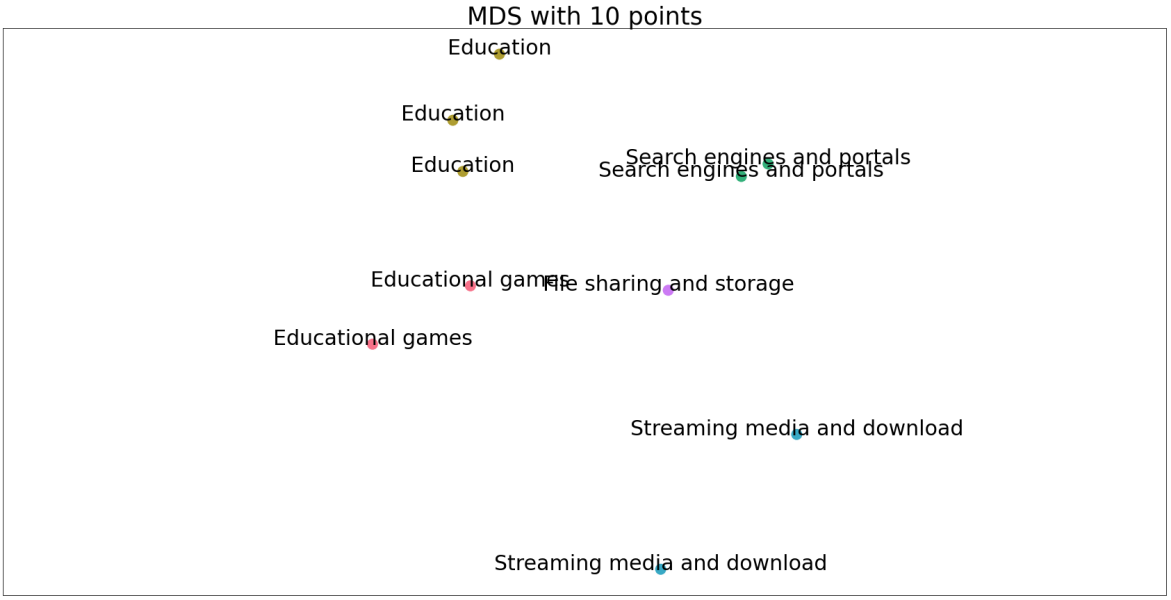

Figure 19: Visualization of 2D Word Embedding using MDS with 10 Points 4

The third embedding we used is t-Distributed Stochastic Neighbor Embedding or t-SNE. The difference between PCA and t-SNE is the fundamental technique they both implement to reduce dimensionality. T-Distributed Stochastic Neighbor Embedding (t-SNE) is an unsupervised, non-linear technique primarily used for visualizing high-dimensional data. T-SNE differs from PCA by preserving only small pairwise distances or local similarities whereas PCA is concerned with preserving large pairwise distances to maximize variance. There are two hyper-parameters that are tuneable (perplexity, n_iter); Perplexity can be explained by the number of close neighbors each point has. The perplexity value has a complex effect on the resulting pictures. More iterations, the more stable it will be. Figure 20 from toward data science shows that the higher the perplexity is the higher the value variance it has.

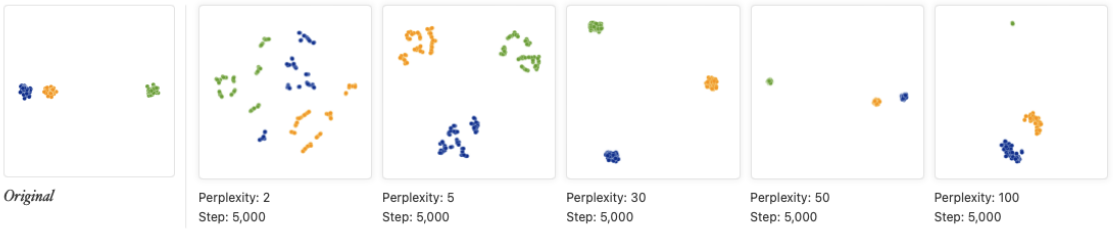

Figure 20: Comparison of perplexity and iteration in T-SNE

For our project, we tested perplexities are 50, 70, and 90 with 2000 iterations in Figure 21. Iteration is fixed because the data points are stable after the 2000 iteration. In other words, the shape of the plot won't change anymore after the 2000 iteration. The way

to tune hyper-parameter perplexities is to visualize and use the category column as a reference to roughly see if it is good enough, also, we can use 1000 manually labeled data to verify. Intuitively, the plot looks underfitting compared with others when the perplexity is 50 and 70. When the perplexity is 90, clearly see that most of the data points are clustered together. When the perplexity is 200, we can see the clusters reducing, which means it combines some of the clusters into one cluster. At this moment, we comprehend that it would not perform well when perplexity is 50 and 70. And we need to experiment when the perplexity is 90 and 200.

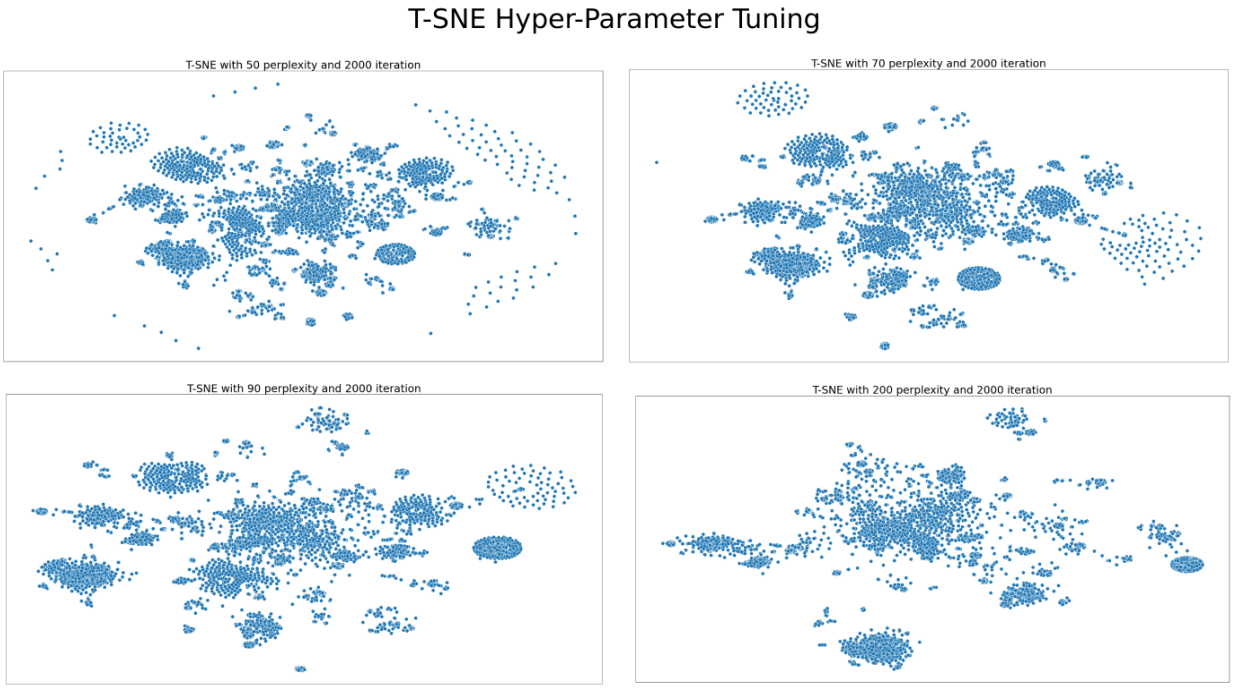

Figure 21: T-SNE Hyper-Parameter Tuning

After the experiments, we found that 200 perplexity messed up the data points (Figure 22 and Figure 23). It is clustered together with some other data points that shouldn't be included, and the bottom cluster is an example that contains other un-correlated categories. In the middle of the plot, there are many data points that it doesn't distinguish well. We can conclude that when perplexity is 90 would perform better compared with perplexity is 200. Even though there are still some of the data points it cannot distinguish, we have more clusters, then we can pick clusters by ourselves to give the final result since one of the goals is as detailed as possible for the embedding method.

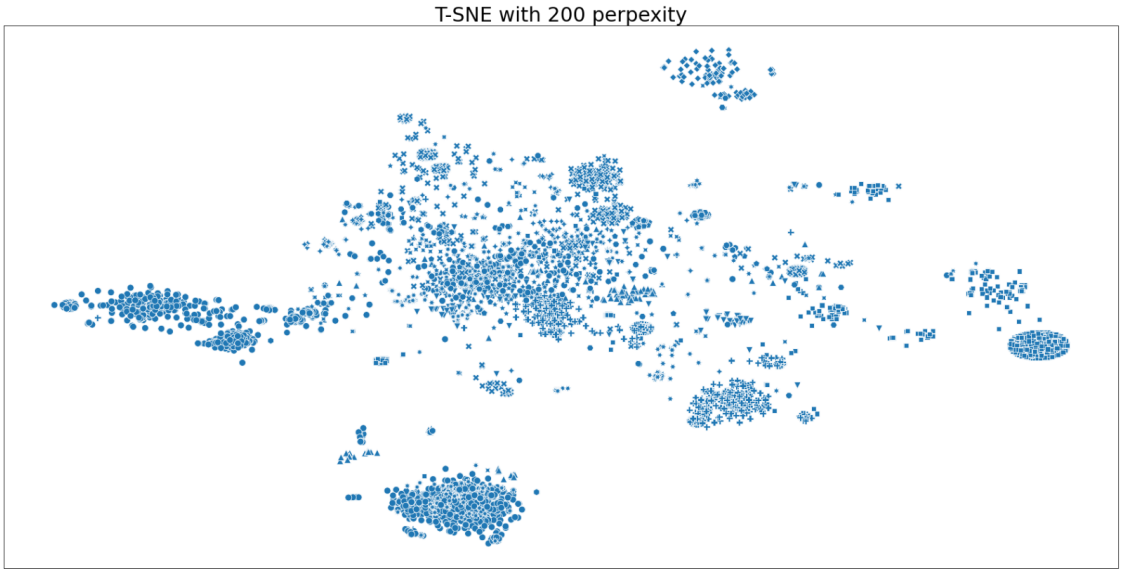

Figure 22: Experiment of T-SNE with 200 Perplexity

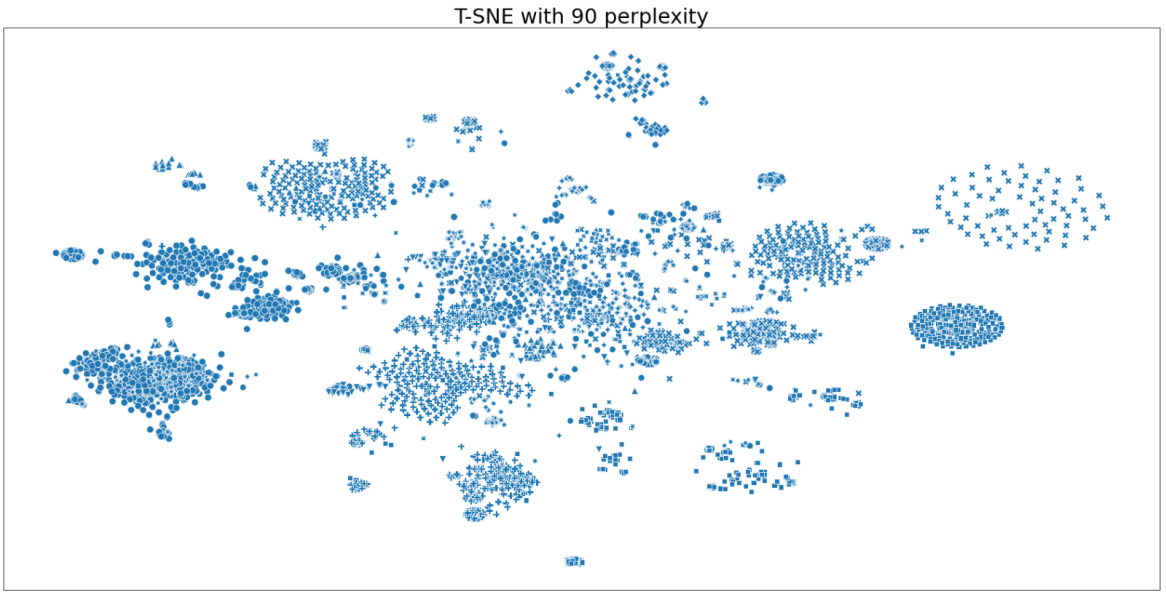

Figure 23: Experiment of T-SNE with 90 Perplexity

We used a couple of sets of 10 points as PCA which we know the true label is to check the integrity or correctness of each embedding method. In Figure 24 to Figure 27 shows that it is like a circle, the education on the right side, and the entertainment on the left side. Google Docs and Google Sheets are a bit closer to the entertainment. And the same titles stick together. However, it can distinguish between different categories of data, which is acceptable.

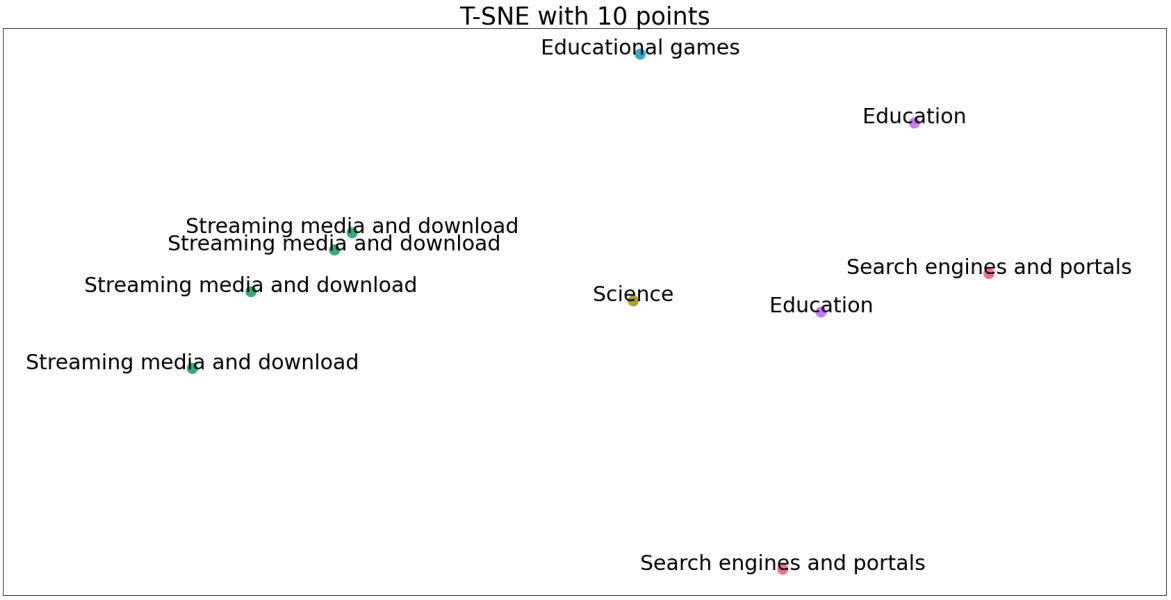

Figure 24: Visualization of 2D Word Embedding using T-SNE with 10 Points 1

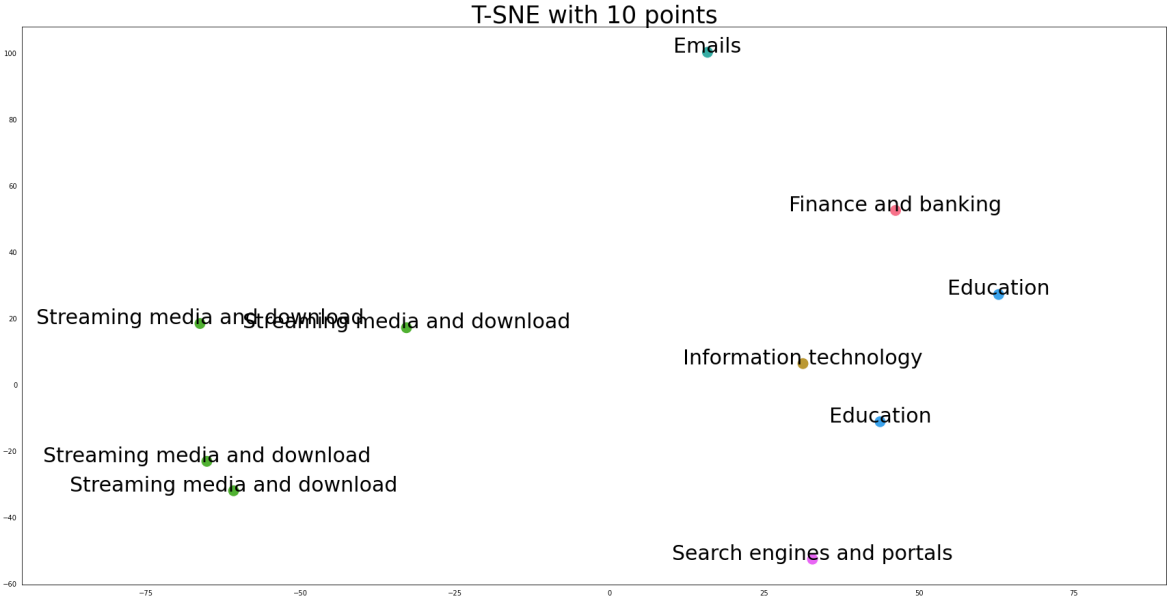

Figure 25: Visualization of 2D Word Embedding using T-SNE with 10 Points 2

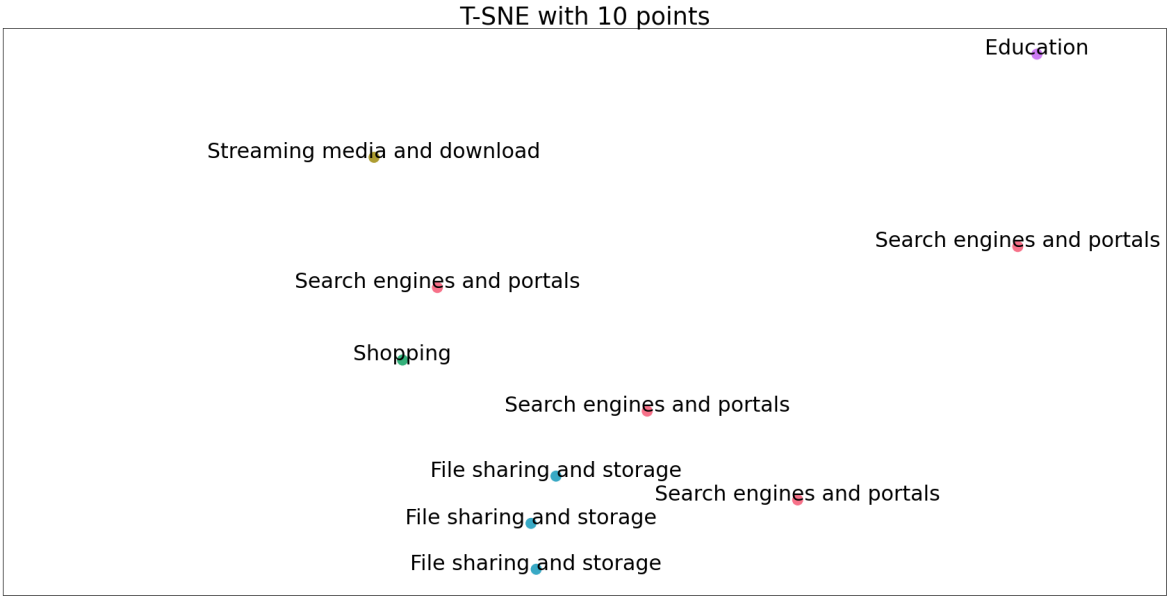

Figure 26: Visualization of 2D Word Embedding using T-SNE with 10 Points 3

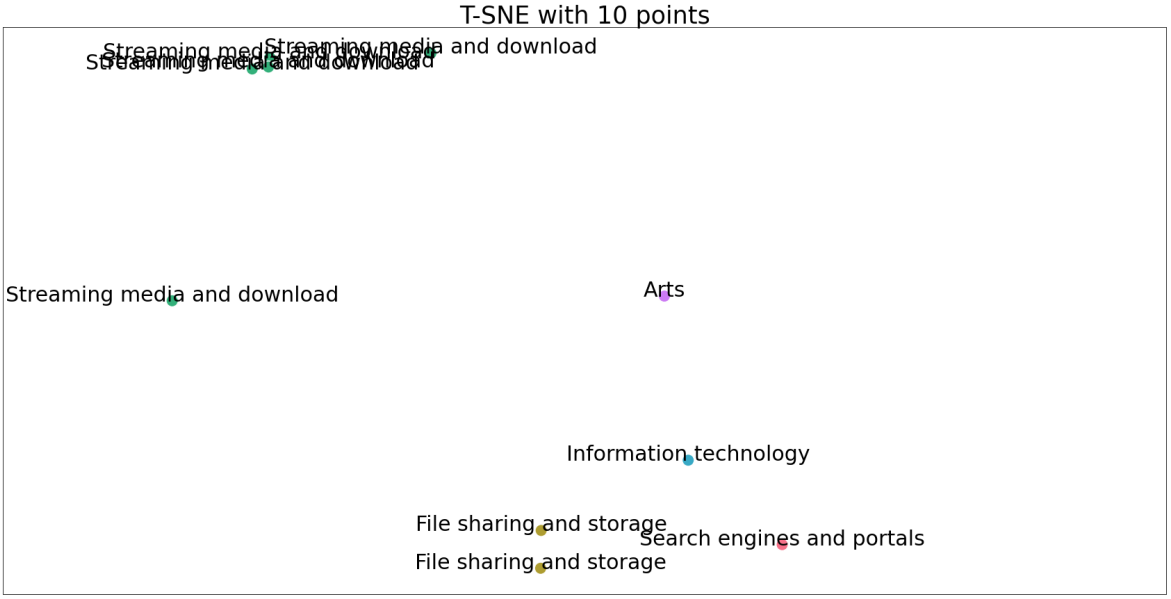

Figure 27: Visualization of 2D Word Embedding using T-SNE with 10 Points 4

The last embedding method we used is Locally Linear Embedding (LLE), which is a Manifold learning(non-linear method) for dimensionality reduction and manifold learning. The LLE procedure has three steps: it builds a neighborhood for each point in the data; finds the weights for linearly approximating the data in that neighborhood, and finally finds the low-dimensional coordinates best reconstructed by those weights. These low-dimensional coordinates are then returned.

There are two major hyper-parameters that can be change:

- *n_neighbors*: Number of neighbors to consider for each point.
- *method*: Four different algorithms
    - Standard: Use the standard locally linear embedding algorithm.
    - Hessian: Use the Hessian eigenmap method. This method requires n_neighbors > n_components * (1 + (n_components + 1) / 2.

- Modified: Use the modified locally linear embedding algorithm.
- Ltsa: Use local tangent space alignment algorithm.

We have tried tuning for *n_neighbors* with different numbers(50, 100, 300, 500) and different parameters for the *method* as well and see if we can see some of the clusters from it. Unforually, we cannot extract the information from LLE(See Figure 28 and Figure 29), the range of x-axis and y-axis is small as well, so it's challenging to cluster. One of the reasons that LLE does not work could be that the Glove word embedding does not work. There are many different word embedding algorithms that capture the semantic similarity of words in a different ways. Another reason could be if the data does not contain the geometric information for LLE to capture, which LLE lost most of the information during the embedding process.

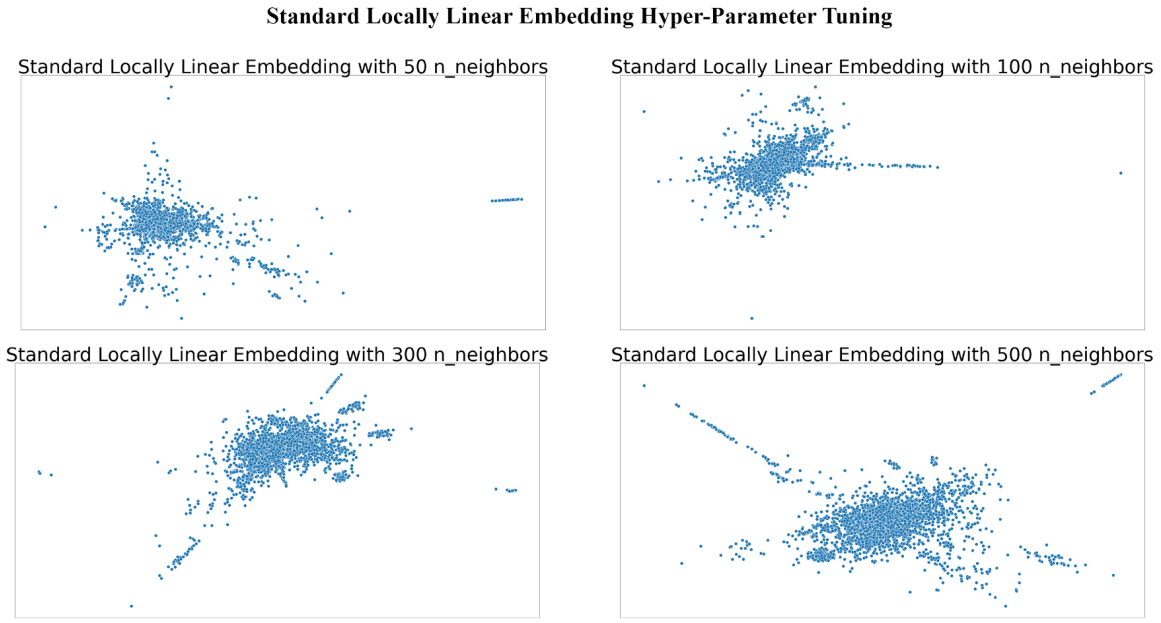

Figure 28: Visualization of 2D Word Embedding using SLLE

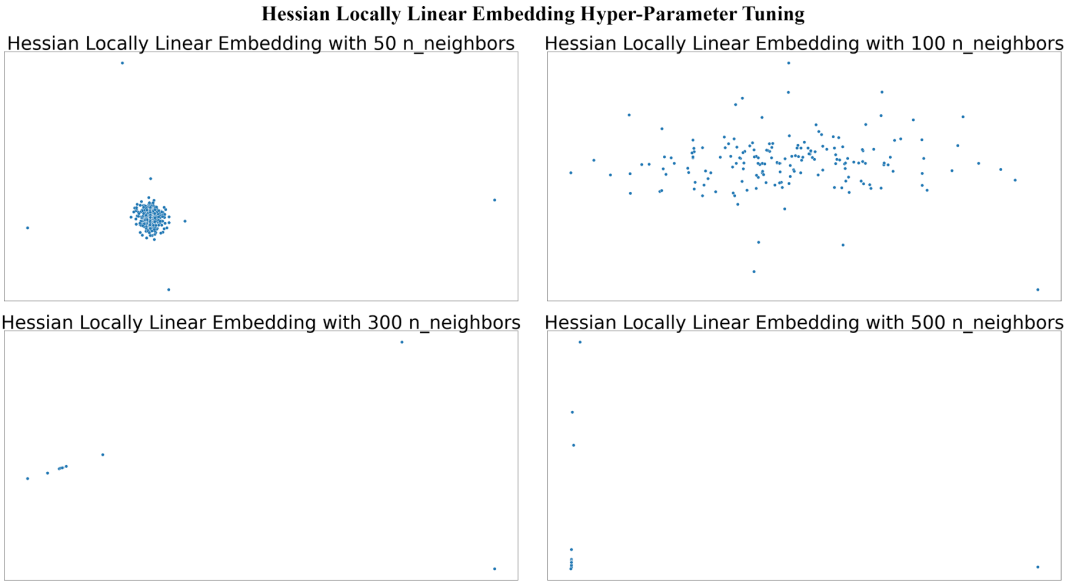

Figure 29: Visualization of 2D Word Embedding using HLLE

The purpose of doing a 2D plot is we want to visualize how good is the embedding method that similar objects should be close to each other. The hypothesis is that in the same cluster similar URLs or titles should be close to each other either in higher dimensional fields or 2 dimensional fields. We expected that the embedding method maintains the closeness between points in that cluster after the embedding. And then we can say the embedding method saves most information. Also, we cannot really compare the different embedding methods because the algorithms are different, the distance between each point is different as well, so the only way to verify the clusters is to visualize them.

Clustering

After we decided on the embedding method, the next step is the apply clustering methods to identify and group similar data points. Selecting an appropriate clustering algorithm for your dataset is often difficult due to the number of choices available. Some important factors that affect the decision include the characteristics of the clusters, the features of the dataset, the number of outliers, and the number of data objects. We did some experiments and decided on three approaches to clustering, K-means(Centroid-based), HDB-scan(Density-based), and Gaussian Mixture Model(Distribution-based). These are very popular clustering methods. The algorithm of these three types of clustering algorithms are distinct, then we will know which type works better. Since it is unsupervised learning, we don't have a quantitative method to evaluate the clustering result, so the ground truth labels are necessary, and we manually labeled the categories for more than 1,000 titles based on URL. And then we can use the ground true labeled data to evaluate the clustering methods.

K-Means

K-means clustering is one of the simplest and most popular unsupervised machine learning algorithms. The goal of K-means is simple, basically is to separate data into k clusters in a way that data points in the same cluster are similar and data points in the different clusters are farther apart. K-means runs over many iterations to converge on a good set of clusters, and cluster assignments can change on each iteration. There is one primary hyper-parameter:

- n_clusters(k): The number of clusters to form as well as the number of centroids to generate.

There are some advantages and disadvantages for K-Means.

K-Means Advantages:

- Relatively simple to implement
- Guarantees convergence
- Scales to large data sets

K-means Disadvantages:

- Difficult to choose k
- Outliers would affect the centroids
- Lack of flexibility in cluster shape

Experimental Analysis and Evaluation For K-Means

For tuning hyper-parameter, we only need to focus on n_clusters(k) will be enough for K-Means. There are several ways of doing it, but the most common way to do is to use the Elbow Method. Elbow Method is an empirical method to find the optimal number of clusters for a dataset. In this method, we pick a range of candidate values of k, then apply K-Means clustering using each of the values of k. Find the average distance of each point in a cluster to its centroid, and represent it in a plot. Pick the value of k, where the average distance falls suddenly. But in our case, when K is greater than certain points, the Elbow method won't work. We can see from Figure 30, that the line is smooth and it's very hard to see where the elbow is. Thus, this is considered a disadvantage for K-means.

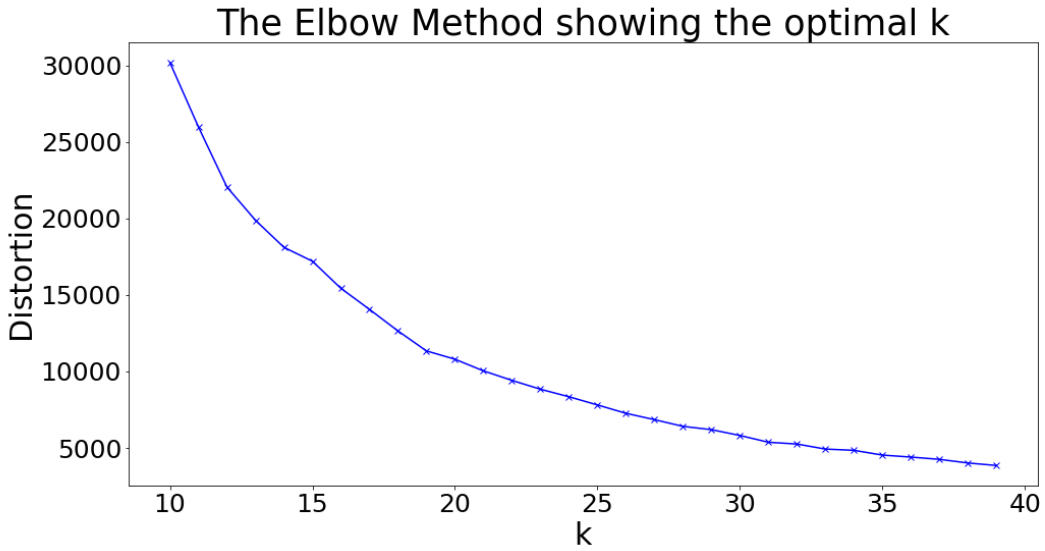

Figure 30: K-Means Elbow Method

However, we can use another method that can determine k is called the Silhouette method. This method is to find the optimal number of clusters and interpretation and validation of consistency with clusters of data. The algorithm is Computed silhouette coefficients for each point and averages it out for all the samples to get the Silhouette score. Then we can use it to tuning parameter k.

The value ranges of the Silhouette score between [-1, 1]. The Silhouette score of 1 means clusters are well and be able to distinguish different clusters. The Silhouette score of 0 means clusters are overlapping, or we could say the distance between each cluster is not significant. The Silhouette score of -1 means clusters are bad, the clusters it assigned were in the wrong way.

Based on Figure 31, we can see that the points after 15 keep increasing, and the highest one after 15 is 28. It starts to fall after 28 and then goes up at 32. There's not much difference between these two numbers, and that is the number of clusters we can choose. As we mentions before, we sampled 30 categories in total. So 28 and 32 make sense. The reason for not choosing the k before 15 is that there is a little off if we choose before 20, and some of the noise may be included in clusters. The K-Means separating a category into two would be better than the K-Means clustering with some other un-correlated categories together. We found that 28 is cleaner compared with 32. Eventually, we decided to use 28 for *n_clusters*.

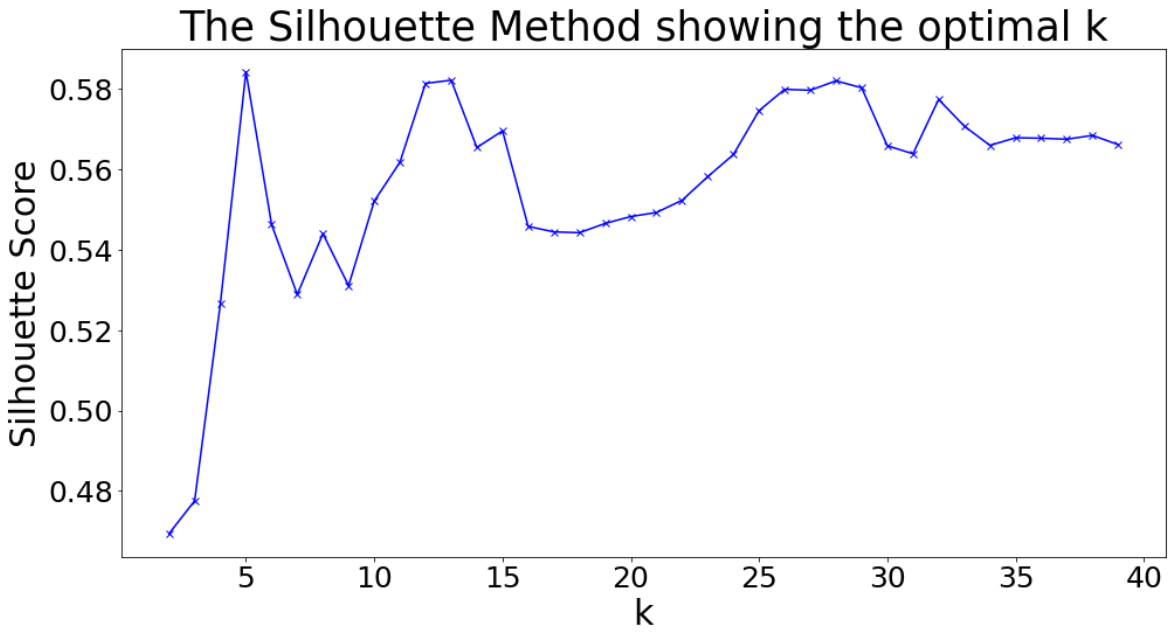

Figure 31: Silhouette Method for K-Means

Gaussian Mixture Model

Gaussian Mixture Model also called GMM, is also a popular clustering method in the machine learning field. As its name implies, each cluster is modeled according to a different Gaussian distribution. This flexible and probabilistic approach to modeling the data means that rather than having hard assignments in clusters like K-Means, we have soft assignments. This means that each data point could have been generated by any of the distributions with a corresponding probability. There are two hyper-parameters we can change:

- n_components: The number of mixture components.
- covariance_type: String describing the type of covariance parameters to use.

There are some advantages and disadvantages of using the Gaussian Mixture Model.

Gaussian Mixture Model Advantages:

- Can handle very oblong clusters
- Involve covariance, which is more flexible for the shape of the distribution
- Provide the probabilities that a given data point belongs to all possible clusters
- Handle overlapped clusters

Gaussian Mixture Model Disadvantages:

- High computational cost
- Required many parameters to fit

Experimental Analysis and Evaluation For GMM

As usual, we have to tune the hyper-parameter for Clustering method. There are two ways to find the best components, which are AIC/BIC and Silhouette Method. We will combine both ways to find out the best *n_components*. We know that GMM is a generative model, so we can evaluate the likelihood function of the model. AIC and BIC can be used to correct the overfitting, adjust the likelihood function, and determine the components for GMM. In order to avoid overfitting, AIC/BIC penalizes the models when the number of clusters goes up. The measurement of AIC/BIC is that the lower the AIC/BIC is, the better the model performs. In Figure 32, we can see that there is a spike on 27, then the 27 can be considered as a n_components. Now, we have a rough idea of what is the n_components. The Silhouette Method shows that 29 is the best n_components should be in Figure 33. We can conclude that the n_components can be either 27 or 29, then we just use 29.

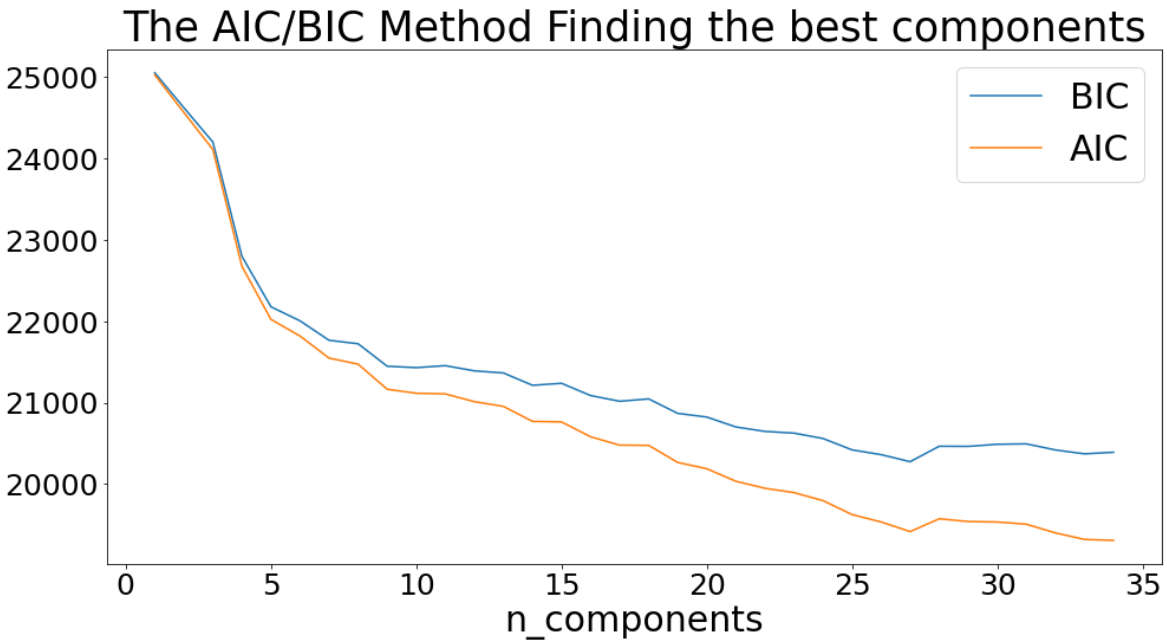

Figure 32: AIC and BIC for GMM

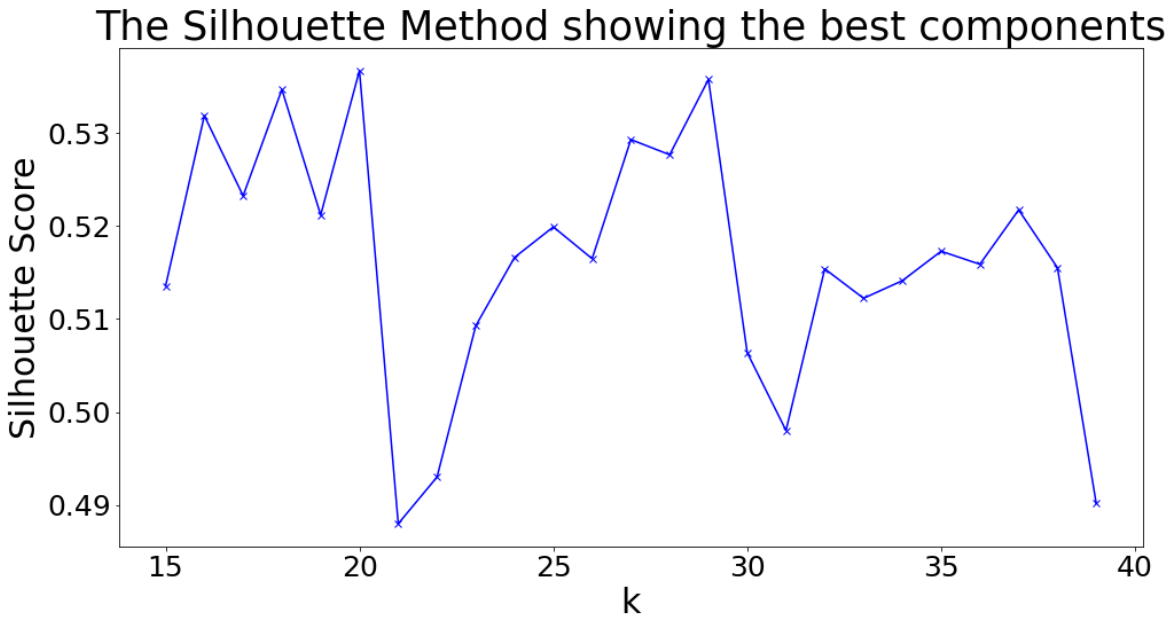

Figure 33: Silhouette Method for GMM

HDBSCAN

HDBSCAN stands for Hierarchical Density-Based Spatial Clustering of Applications with Noise. It extends DBSCAN by converting it into a hierarchical clustering algorithm and then using a technique to extract a flat clustering based on the stability of clusters. We noticed that the HDBSCAN makes only a single pass through the data, and once a point has been assigned to a particular cluster, it never changes. The performance is similar to DBSCAN but the tuning hyper-parameter will be easier than DBSCAN since we cannot find a way to evaluate whether the epsilon and min_points are good enough. There are 3 main hyper-parameters:

- min_cluster_size: It is a primary parameter to affect the resulting clustering and it is a relatively intuitive parameter to select.
- min_samples: Similar to min_points in DBSCAN. The larger the value of min_samples is, the more conservative the clustering. In other words, there are more points will be declared as noise, and clusters will be restricted to progressively more dense areas.

Experimental Analysis and Evaluation For HDBSCAN

We noticed that the min_cluster_size is a primary parameter based on documentation. After testing some of the min_samples equal to 1 with fixed min_cluster_size, the clustering didn't make huge changes. We fixed the min_samples and change min_cluster_size with different numbers. The way of determining min_cluster_size is by using the built-in function *relataive_vaildity_,* also called the Density-Based Cluster Validity (DBCV) score. This is an objective score between -1 and 1 on the quality of

clustering, The score of 0 means that clusters are overlapping, the score of fewer than 0 means that data belonging to clusters may be incorrect, and the score of 1 means that clusters are correct or performs well. We start from 50 and increase by 10 each time until 170 in Figure 34. We can see that after min_cluster_size is equal to 150, the score starts to decrease, which means the HDBSCAN starts to overfit after 150. Thus, the best min_cluster_size we can have is 150.

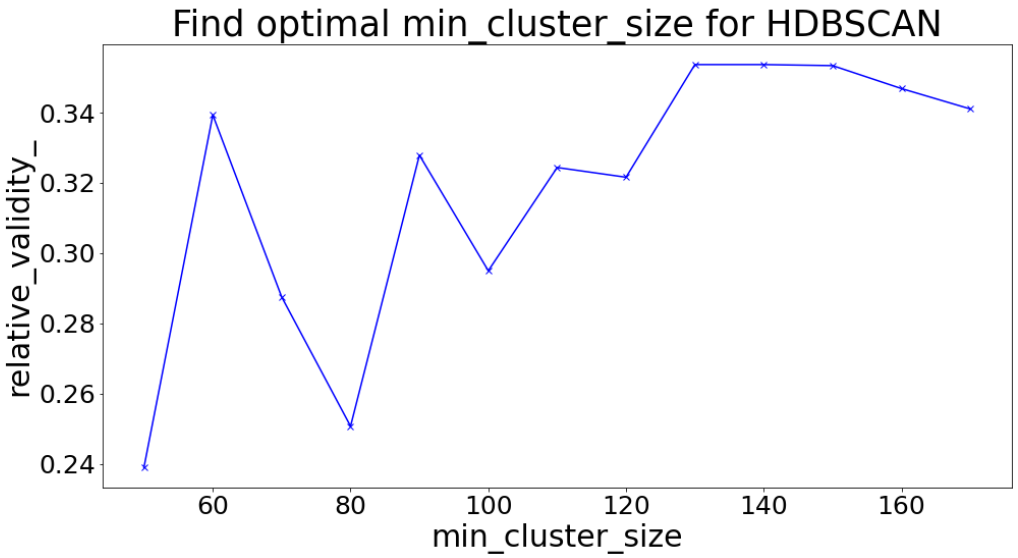

Figure 34: Hyperparameter tuning for min_cluster_size in HDBSCAN

Clustering Results

We have everything now, then we can visualize what is the clustering result for K-Means. There are some data points that either T-SNE or K-Means cannot really distinguish, which we can consider as noise because the data set is imbalanced, so we don't have enough data to group them. We are going to manually look through each cluster and see whether the clusters make sense or not. First, we can use the hard label as a reference to roughly see the distribution of each cluster. The first column of the Table… is the cluster number, the second column is the categories in this cluster, the third column is how many categories are in this cluster, and the last column is how many data points are in the cluster. We notice that cluster 15 contains the most data. Then we did some analysis of cluster 15, the distribution of cluster 15(See Figure 36), we can see that it's mostly from the "Streaming media and download" category. That means the clustering method is grouping similar titles. And then we sampled 20 points out and the table(See Table 4) shows that this cluster is related to sports, because most of the titles are about basketball games on YouTube. This is an example of how to determine the category of the cluster.

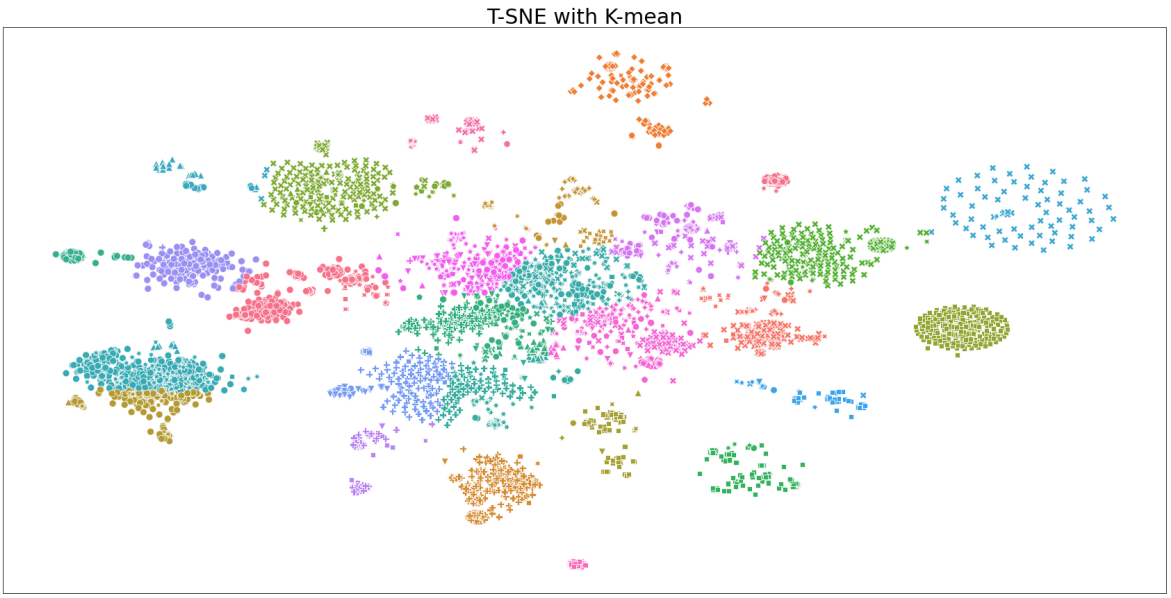

Figure 35: T-SNE with K-Means

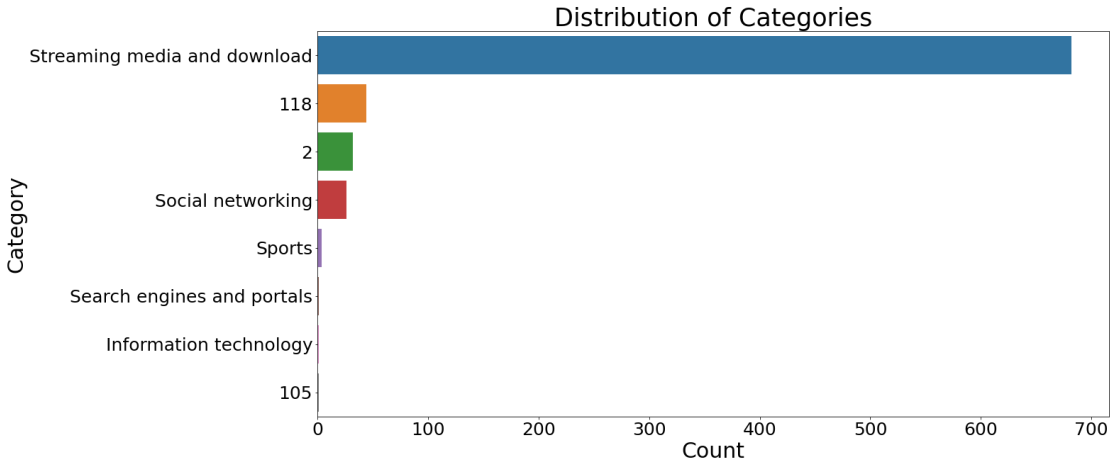

Figure 36: Distribution of cluster 15

category	cluster	title
Streaming media and download	15	2019 NCAA Wrestling (133 lb) Quarterfinal - (2...
Streaming media and download	15	Ohio State's Justin Fields throws 6 TDs in Sug...
Streaming media and download	15	2019 NCAA Wrestling (125 lb) Championship Rd.2...
Streaming media and download	15	2019 NCAA Wrestling (133 lb) Quarterfinal - (2...
Streaming media and download	15	(423) do the roar.mp4 - YouTube
Streaming media and download	15	(172) All Sports Golf Battle | Dude Perfect - ...
Streaming media and download	15	2020 NBA 3-Point Contest - Full Highlights | 2...
Streaming media and download	15	NUGGETS at TIMBERWOLVES | FULL GAME HIGHLIGHTS...
Streaming media and download	15	NEW Thomas Sanders Vine Compilation | THOMAS S...
Streaming media and download	15	NFL Embarrassing Fails of the 2020 Playoffs - ...
Streaming media and download	15	mikey williams overtime episode 2 - YouTube
Streaming media and download	15	Falcons vs. Buccaneers Week 17 Highlights | NF...
2	15	Top 20 NFL Teams With MOST Conference Champion...
Streaming media and download	15	Olympic Wrestling Trials | Kyle Dake vs J`Den ...
Streaming media and download	15	(13) Hanson GB vs Gregory - YouTube
Streaming media and download	15	2019 NCAA Wrestling (197 lb) Championship Rd.2...
Streaming media and download	15	PACERS at PELICANS | FULL GAME HIGHLIGHTS | Ja...
Streaming media and download	15	LAKERS at GRIZZLIES | FULL GAME HIGHLIGHTS | J...
Streaming media and download	15	(13) Hanson GB vs Gregory - YouTube
Streaming media and download	15	CLIPPERS at SUNS | FULL GAME HIGHLIGHTS | Janu...

Table 4: Table of random sample in cluster 15

The results are not surprising because the basic algorithm of K-Means and GMM are similar. Intuitively, GMM should be better than K-Means. The difference is how the GMM and the K-Means handle the center part, which is the noise. Compared with Figure 35, We found that GMM actually clustered the noise together(Green color and blue), and release the red part, and the K-Means clustered that part with the noise. And rest of the clusters are just very similar.

We know that HDBSCAN or DBSCAN can cluster different shapes of the clusters. So it successfully clustered most of the points around the plot. However, it has difficulties handling the center part, and that is a disadvantage of HDBSCAM, once the data points overlapping, it is challenging for HDBSCAN to cluster.

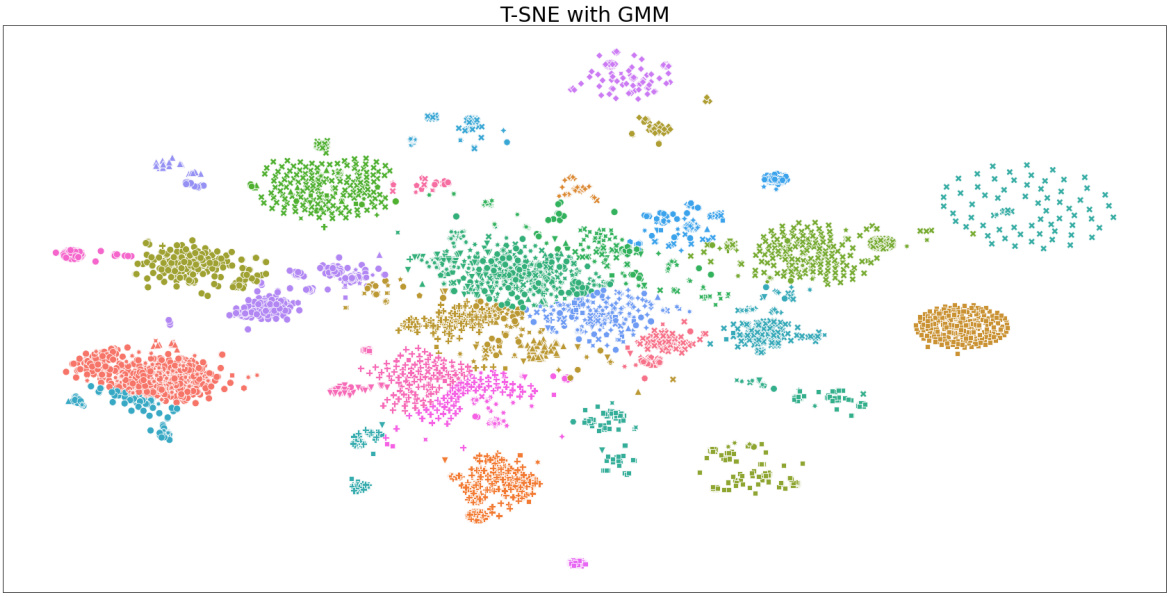

Figure 37: T-SNE with GMM

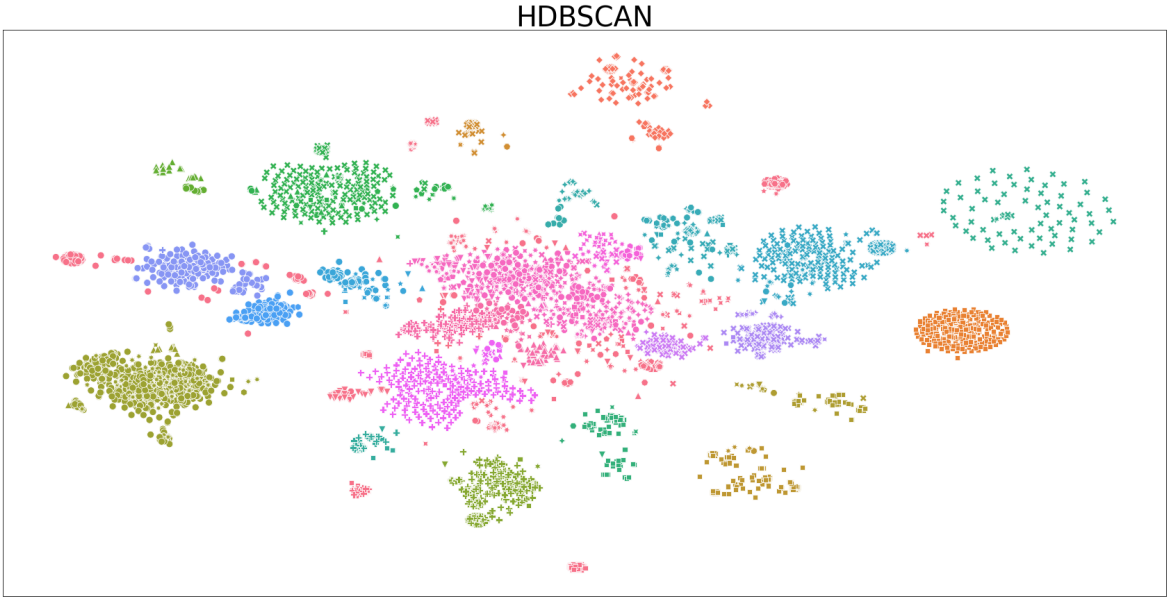

Figure 38: T-SNE with HDBSCAN

Results

Precision, Recall	K-Means	GMM	HDBSCAN
Precision	0.6191	0.5668	0.6161
Recall	0.7371	0.4412	0.7546
F1 Score	0.6543	0.496	0.6783

Table 5: Precision, Recall and F1 score Evaluation

Based on the result we have done and found that HDBSCAN gives the best dimension reduction, and HDBSCAN gives the best F1 Score (See Table 5). We expected the GMM performance should be better than K-Means intuitively, but in our case, the GMM actually performed very poor. And the HDBSCAN surprisedly performed not bad since we did spend a lot of time on hyper-parameter tuning for DBSCAN and just have no idea how to verify it, and then we change to HDBSCAN.

However, for some points, the clustering method does not capture it because the data points for those categories are not balanced. However, we'll focus more on the major categories which have more data points on them. For example, Education, social networking, streaming media and download, etc…

Finally, we want to create the word cloud to present our clusters to the either parent or the teacher. We are very interested in the most common words that show in the major categories, which we can get from the clusters we just found. There are three very interesting clusters, which can represent student behavior. The Biggest word is 'Youtube' for the first word cloud, so this cluster should be a streaming media and download cluster. We can also see some words about sport, such as 'Football', 'Basketball', 'NBA' and 'Highlight' etc… Thus, the students are more interested in football and basketball in the sports category. (See Figure 39)

Figure 39: Word cloud for Streaming Media and Downloaded in Sports

In Figure 40 is also considered a Streaming Media and Download cluster, and most words are 'Instagram', 'Stories' and 'Pinterest', etc… The word 'Instagram' and 'Stories' are together, that's why they are the same size. The students mainly visit Instagram for social media. (Figure 40)

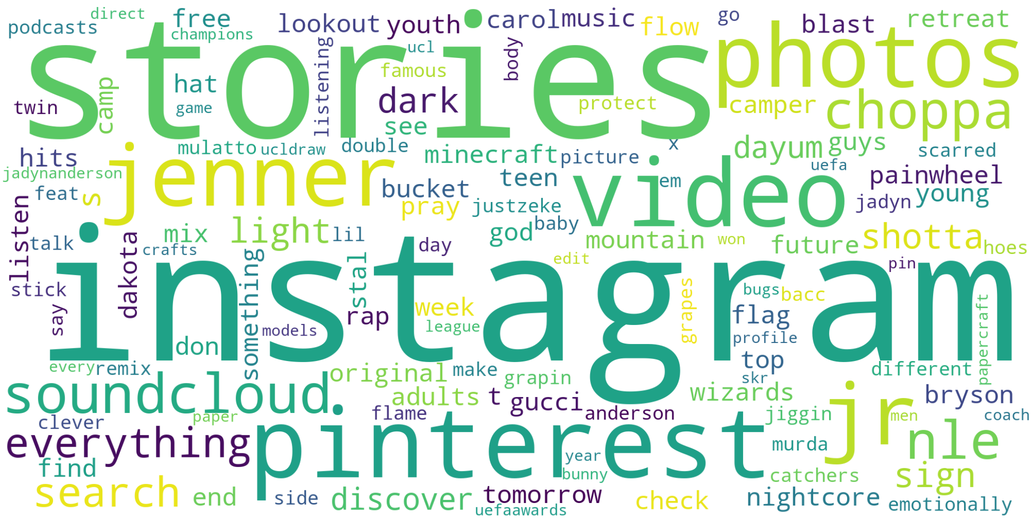

Figure 40: Word cloud for Social Networking

In Figure 41 seems like an Education category. The most common word ‘Realize Savvas’ is the website that allows students to access their courses and groups, in which students can submit assignments, group discussions, or view course materials. The word ‘Edgenuity’ is also a website for students taking courses online. Also, some of the other words are interesting, such as ‘Flashcards’, ‘Quizlet’ and ‘Science’, etc… Most of the words look related to education.

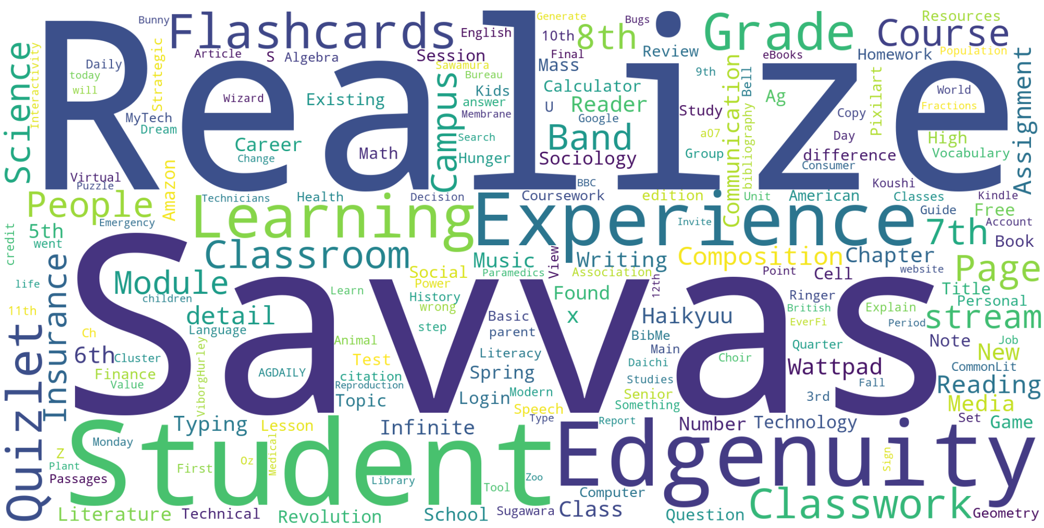

Figure 41: Word cloud for Education

Conclusion/ Discussion

In conclusion, the project analyzes the student browsing behavior patterns and uses interactive visualization to better understand students' activities. This project used four different embedding methods and three different clustering algorithms to achieve our goal and explained each method and algorithm's performance. It is a good opportunity to learn about unsupervised learning and different methods to handle word similarity projects in the real world. If we have more time, we can improve on the sampling, because compared with the original data it is still too small. Some of the titles are not representative, and the solution we can think of is that we can grab more information from the website instead of only titles and URLs, such as the articles and pictures. There are many other different word embedding methods, such as Embedding Layer, Word2Vec, Tweet, etc… That we can try to see if it will be better than Glove word embedding or not. We can also improve the titles we have right now. For example, we can translate Non-English words into English words. Also, we can make the categories balance, then we will have better clusters for each category.