A Clustering System For YouTube Channel Classification

Bradley Huang brad.huang@nyu.edu
Shenyi Huang sh6210@nyu.edu
Deniz Qian dq2024@nyu.edu
Daniel Ni dn2212@nyu.edu

Abstract

In this paper, we propose an unsupervised YouTube channel classification system based on the content of the channels. We extract different numbers of keywords and their weights from the video scripts of 225 channels and use the Word2Vec technique to generate embeddings for each keyword. We generate clusters with the similarity scores between all channels with different threshold values. We conduct quantitative analysis and qualitative analysis based on a manually tagged label baseline system and a predefined YouTube label baseline system. Finally, our analysis shows that our system has the best performance when more keywords are used for each channel and a lower threshold value is applied when calculating the similarity score.

1 Introduction

Online video sharing and streaming platforms have become a large part of the 21st century. YouTube, being one of the leading platforms, has grown exponentially since its initial launch in 2005. The platform is home to millions of channels that produce content that covers a wide array of categories. As the platform continues to grow, users often struggle to find channels that align with their interests due to a mass amount of videos covering various topics. Upon creation of a new YouTube channel, content creators are tasked with choosing one of 15 categories to describe the type of content they will publish. However, many of them steer away from this initial choice. This causes their newer videos to be filtered into incorrect categories on third party platforms that users use to search for new content creators. This problem calls for an efficient and scalable solution to categorize YouTube channels in order to improve a user experience with regards to content discovery.

In this project, we developed a system to cluster similar YouTube channels based on a channel's most popular videos' transcripts. By organizing the channels into clusters, our project enables users to avoid surfing through millions of channels to find content, but rather can observe a cluster that suits their interests and find new content creators from it.

Our primary contributions are as follows:

- We describe a general framework for our YouTube channel categorization that relies on multiple methodologies and implementations (see section 3).
- We assess the quality of our clusters by providing a qualitative analysis of our final graphs by comparing different graphs that were created using different numbers of keywords and different thresholds of word similarity.
- We also assess the quality of our clusters by providing a quantitative analysis using different metrics such as modularity and F-score which are calculated by comparing the channels in the clusters created by our system to the predefined data set and manually tagged data set.

The paper is organized as follows. In Section 2, we introduce related work pertaining to content-based classification and clustering techniques. In Section 3, we describe in detail our methods and implementations for YouTube channel categorization. In Section 4, we describe an evaluation of our results, including both a quantitative and qualitative analysis. In Section 5, we draw final conclusions and talk about future work that can be done for our project.

2 Related Work

Manual keyword extraction is highly cumbersome and, there is much research in developing unsu-

pervised keyword extraction from texts. Unsupervised methods can be categorized into graph-based, statistic-based, and topic-based methods (Papagiannopoulou and Tsoumakas, 2019). Within topic-based KPE methods, there are clustering-based methods where candidate keyphrases extracted from text are agglomerated and used as vertices in a complete graph and given a significance score with a graph-based ranking model (Bougouin et al., 2013). Finally, key phrases are then selected from candidates in top-ranked topics and returned. KeyBERT utilizes an unsupervised KPE model denoted as Masked Document Embedding Rank, in which it employs a novel method of 'masking' keywords in a hence masked version of the original document and looking for the least semantically similar sections to pick out keywords(Zhang et al., 2023). The logic in this is that the most important keywords are unique and thus should near completely define the meaning of sentence structures in the document, meaning that their removal should cause the greatest discrepancies in semantic similarities when comparing the masked document and the original document. TextRank is a graph-based model which applies an adapted version of Google's PageRank algorithm(Brin and Page, 1998) iteratively on a graphical representation of the inputted text to rank and extract keywords top, similarly to how the PageRank algorithm does so to rank the relevance of websites. TextRank introduces the idea of 'recommendation', where importance of words in a text is identified not only through local context of a single graph unit, but also by recursively computing 'recommendations' given by other related text units in the graph, and finally computing an importance score based on how important its related words are. TextRank follows PageRank's idea of the 'random surfer model', where the reader is likely to follow strings of words to find concepts that have relation to the current concept that is being read.

3 Methods and Implementation

In this section we discuss our six different methods for accomplishing our goal in chronological order of implementation: Data Selection, Data Scraping, Keyword Extraction, Similarity Score Calculation, Clustering Technique, and Modularity. We used a variety of different algorithms and python libraries to implement these methods. In the following, we dive into a detailed description of these implementations.

3.1 Baseline Systems and Data Selection

In order to evaluate our system, we created two baseline systems to compare our final clusters to: the categories that were defined by YouTube that the YouTubers selected themselves upon creation of their channel and the categories that were annotated by us based on the YouTubers' content.

The first baseline system is a list of 15 channels from 15 categories, a total of 225 youtube channels. The 15 categories are YouTube's default categories that all YouTubers need to select when creating their channels: Autos & Vehicles, Comedy, Education, Entertainment, Film and Animation, Gaming, Howto & Style, Music, News & Politics, Nonprofits & Activism, People & Blogs, Pets & Animals, Science & Technology, Sports, and Travel & Events. We did not include channels that did not provide transcripts due to the fact that it would be impossible to scrape data from these channels. We also only took into account English speaking channels and skipped channels that mainly spoke in a foreign language since our system can only process transcripts in English.

We then used Social Blade to sort the most subscribed YouTube channels in descending order for each of the 15 categories. Social Blade is a leading online platform that provides analytics and statistics for YouTube channels. The platform provides users with insight into a wide variety of metrics such as subscriber count, view count, and estimated earnings. By analyzing the data from millions of YouTube channels, Social Blade is able to provide a ranking list of top channels across various categories. After sorting the channels in each category, we then manually compiled a data set of 225 youtubers, their channels link, and their default labels.

Our second baseline continued off of our already created data set of 225 youtubers based on Social Blade's ranking system. Using the same list of YouTubers, 4 annotators manually categorized each of the 225 channels based on the content of each channel. In total we had 18 manually assigned categories that are as follows: Cars, Variety Entertainment, Comedy, Technology, Food, Education, Kids, Beauty, Film, Gaming, Howto, Music, News, Politics, Religion, Animals, Sports, and Travel. After compiling both of our baseline data sets, we then were able to move onto scraping data from the channels.

3.2 Data Scraping

Our overall goal for data scraping was to scrape the transcripts of the top 10 most viewed youtube videos with English transcripts for each of the YouTubers in our data set (see 3.1), merge the transcripts of the 10 videos to build a corpus for each YouTuber, and output all YouTubers' names and corpora into a single JSON file.

To accomplish this goal, we determined each channel's top 10 most viewed videos by using the *scrapetube* library which returned to us 10 video IDs for each channel. With these video IDs, *YouTube Transcript/Subtitle API* was used to scrape the transcripts of the videos corresponding to the video IDs. All of the scraped data was formatted using the text formatter provided by the API and stored into our corpus. With the scripts corpus, we were able to move on to keyword extraction to provide a more concise categorization for each channel.

3.3 Keyword Extraction

BERT, also known as Bidirectional Encoder Representations from Transformers, is a pre-trained deep learning model for generating word embeddings developed by researchers at Google AI Language(Devlin et al., 2019). With *BERT*, a wide variety of tasks can be performed, such as sentiment analysis, entity recognition and text classification.

In our project, we used *KeyBERT*, a keyword extraction model based on *BERT* word embeddings, to extract keywords from each YouTuber's script. *KeyBERT* utilizes n-gram word embeddings generated by *BERT* and calculates cosine similarity that shows how relevant each word is to the script in order to generate keywords and their weights. This weight represents how relevant a keyword or a keyphrase is in relation to the text. The higher the weight, the more relevant that word is in capturing the overall topic of a text. The YouTube transcripts scraped from 3.2 were used to get a list of single keywords that best represent the topics YouTubers talk about and the weights of the keywords. We will later cluster YouTube channels together based on the similarity of keywords between channels, as well as a weight for each keyword. We used *KeyBERT* to extract 5, 10, 15, and 20 keywords for each channel to provide four different metrics for each channel. In the evaluation part, we later discuss how we used these metrics to evaluate which number of keywords provides the most accurate final clustering graph.

3.4 Similarity Score Calculation

In this section, we describe our approach to calculate a similarity score between two YouTube channels. The process of generating a similarity score was composed of two tasks: generating word embeddings of the keywords of the two YouTubers that will be compared and calculating the similarity score based on the word embeddings and the weights of each keyword.

A word embedding of a word is the numeric vector representation of the word. The more semantically similar two words are, the more similar their word embedding vectors are. One of the best known examples is taking the vectors for the words "man", "woman", and "king", the most similar vector to the result of $vec("king") - vec("man") + vec("woman")$ will be the vector of the word "queen" (Mikolov et al., 2013). In our research, we used a word embedding generating algorithm named Word2Vec.

3.4.1 Word2Vec

Word2Vec is a neural network model that is used to generate word embeddings based on the semantic and syntactic relationships between words (Mikolov et al., 2013). In our research, we used gensim, an open-source Python library that provides tools for natural language processing and information retrieval, to load a pre-trained word and phrase Word2Vec model derived from Google News dataset, which contains 300-dimensional vectors for 3 million words and phrases. With this model, we were able to input a word and get its corresponding embedding.

3.4.2 Weighted Cosine Similarity Score

In order to calculate how similar two keywords are, we first calculated a cosine similarity score between the two keyword embeddings. Then, using the cosine similarity score and the weights of the two keywords generated by *KeyBERT*, we were able to calculate a weighted similarity score.

Cosine similarity (1) is a metric that is used to measure the similarity between two non-zero vectors.

$$S(A, B) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2 \sum_{i=1}^{n} B_i^2}} \qquad (1)$$

Where A and B are the word embedding vectors, and n is the length of the word embedding.

A cosine similarity score ranges from -1 to 1. We have 4 thresholds for the cosine similarity score that determine if the score will be used to calculate the weighted cosine similarity score: 0.6, 0.7, 0.8, and 0.9. We will later evaluate the result obtained by using the four thresholds.

Once we calculated the cosine similarity scores between all the keywords for the two YouTubers that are comparing, we calculated the final weighted cosine similarity score of any two YouTube channels as in (2)(3).

$$TotalWCS = \frac{1}{n}\sum_{i=0}^{n}\sum_{j=0}^{n} WCS_{i,j} \quad (2)$$

Where

$$WCS_{i,j} = \frac{S_{i,j} - \min(S)}{\max(S) - \min(S)} W_i W_j \quad (3)$$

i stands for the index of the keyword in keyword vector 1 (YouTuber 1), and j stands for the index of the keyword in keyword vector 2 (YouTuber 2). $S_{i,j}$ stands for the similarity score between keyword i and keyword j, and S stands for the list of similarity scores between two keyword vectors, and W_i stands for the weight of keyword i, and W_j stands for the weight of keyword j. Notably, when $max(S) - min(S)$ equals 0, or $S_{i,j}$ is lower than the threshold, $WCS_{i,j}$ equals 0. n stands for the count of non-zero $WCS_{i,j}$. If n equals 0, $WCS_{i,j}$ will also equal 0.

3.5 Graphs

To cluster the YouTubers, we created graphs where nodes are the YouTubers, and the edge weights between our nodes are the weighted similarity scores. We used the *NetWorkX* library to generate the graphs. *NetWorkX* is a Python library that helps visualize and manipulate data within networks and graphs. With the graphs, we were able to cluster YouTubers and conduct both qualitative and quantitative analysis.

3.6 Clustering Technique and Modularity

To find communities based on given weights between nodes, we used the $greedy\,modularity\,comunities$ function from *NetWorkX*(Clauset et al., 2004). This function applies the Clauset-Newman-Moore greedy modularity maximization method, which is widely used for larger network analysis in a relatively fast runtime. The function returns a set of clustered communities based on the weights of edges between nodes. Using this set, we can then calculate a modularity score of our graphs with the provided modularity calculation method in *NetWorkX*. A modularity score defines how well separated communities are from each other and how dense these communities are. We then can conduct a quantitative analysis on our system based on the modularity score and how similar our clusters are to two baseline systems.

		Keyword Number			
		5	10	15	20
Threshold	0.6	0.312	0.397	0.46	0.465
	0.7	0.243	0.37	0.392	0.432
	0.8	0.246	0.346	0.374	0.442
	0.9	0.216	0.326	0.355	0.387

Table 1: Manually Tagged List Threshold vs Keyword Numbers

		Keyword Number			
		5	10	15	20
Threshold	0.6	0.287	0.352	0.417	0.423
	0.7	0.21	0.329	0.347	0.394
	0.8	0.206	0.305	0.341	0.396
	0.9	0.182	0.294	0.322	0.341

Table 2: Predefined List F-score vs Threshold

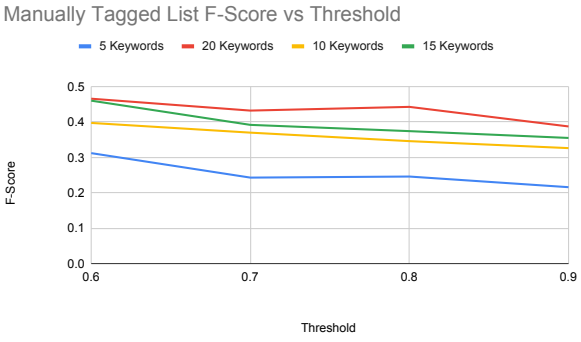

Figure 1: Manually Tagged List F-score vs Threshold

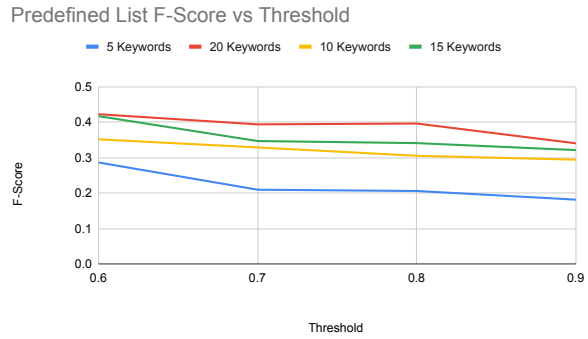

Figure 2: Predefined List F-score vs Threshold

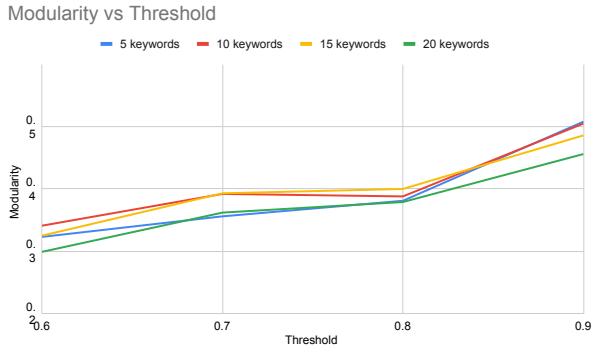

Figure 3: Modularity VS Threshold

4 Evaluation and Results

4.1 F-score Evaluation Method

To calculate our recall, precision, and F-score, we employed a greedy algorithm that compared every single cluster from the predefined list of clusters or manual list of clusters against the clusters formed by our algorithms. This was used so that we can match the clusters with the most intersections with each other to calculate the true positive, false negative, and false positive as pairs for a total of n pairs. We then sum the true positive (tp_i), false negative (fn_i), and false positive (fp_i) from each cluster-to-cluster comparison pair. Following that we calculate the precision, recall, and f-score. This process is repeated for each x keyword and y threshold graph we generate.

$$Precision = \frac{\sum_i^n tp_i}{\sum_i^n tp_i + \sum_i^n fp_i} \quad (4)$$

$$Recall = \frac{\sum_i^n tp_i}{\sum_i^n tp_i + \sum_i^n fn_i} \quad (5)$$

$$Fscore = \frac{2 * P * R}{P + R} \quad (6)$$

4.2 Quantitative Analysis

Table 1 shows the F-scores of our system with different thresholds and keyword numbers compared against the manually tagged YouTuber label baseline system. The system achieved the highest F-score of 0.465 at 0.6 threshold and 20 keywords, while the lowest F-score 0.216 was at 0.9 threshold and 5 keywords. Table 2 shows the F-scores of our system with different thresholds and keyword numbers compared against the predefined YouTuber label baseline system. The system achieved the highest F-score of 0.423 at 0.6 threshold and 20 keywords, while the lowest F-score 0.182 was at 0.9 threshold and 5 keywords.

Comparing the results of the two baseline systems, it was found that our system has a slightly better performance when compared against the manually tagged YouTuber label baseline system. On average, the F-score is 12.498% higher than the predefined YouTuber label baseline system. A possible interpretation of this result is that the predefined YouTuber label for a YouTube channel was defined upon the creation of the channel, and it might not correctly represent the actual topic that most videos in that channel are talking about. Since the manually tagged baseline system is based on the overall content of a channel, and our system used the scripts of the top 10 videos of the channel, the result generated by our system was favored more by the manually tagged baseline system than the predefined label baseline system.

Figure 1 and Figure 2 visualizes the trends of how F-score changes when keyword numbers and threshold changes. In both figures, the more keywords, the better performance the system has; the lower the threshold, the higher the F-score is. One possible explanation for the trend of the keyword is that using more keywords can result in having more words that have different meanings, so the keyword list can better capture the topic a YouTuber is talking about and thus more likely makes a YouTuber connect to other YouTuber with similar topics. For example, Khan Academy is an educational YouTube channel. The top five keywords of Khan Academy are “tutoring”, “tutor”, “classroom”, “classrooms”, and “students”. It can be identified that “tutoring” and “tutor” are very similar in terms of their meanings, same for “classroom” and “classrooms”. Although these words are very relevant to education, more varieties of words are needed to connect this channel to other

educational channels who might not have these keywords in their top five keywords list. A potential reason why a higher threshold results in a lower F-score is because when calculating the weighted similarity score, a high threshold will lead to an overall high weighted similarity score between any two YouTubers since only high cosine similarity scores between two keyword embeddings will be added.

As Figure 3 shows, the higher the threshold, the higher the modularity score, and the modularity score and keyword numbers do not have a significant correlation. A high modularity score indicates that the clustering algorithm has successfully identified groups of nodes(communities) that are more tightly connected to each other than to nodes in other clusters. Therefore, a graph with a high threshold has more well-defined clusters. However, the trend of the modularity score against threshold was the opposite as the F-score against threshold. Thus, we can conclude that a graph with a high threshold has well-separated clusters but with low accuracy, and a graph with a high keyword number is more accurate to represent the categories of YouTubers but the clusters are less well-defined.

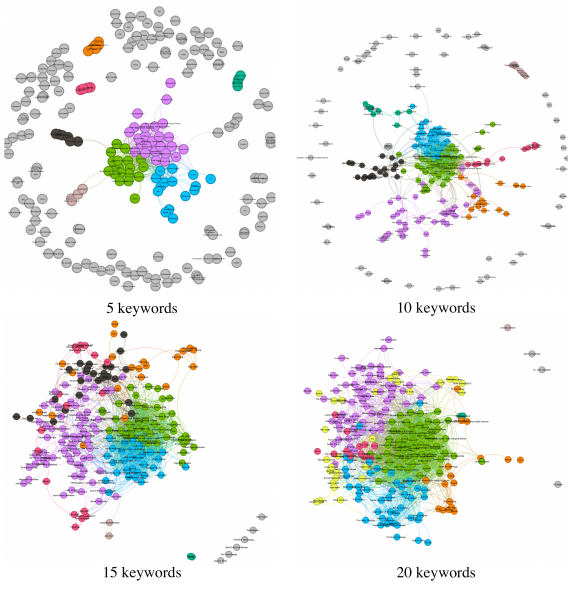

Figure 4: Graphs of 0.6 Threshold with 5, 10, 15, and 20 Keywords

4.3 Qualitative Analysis

We then analyzed the final clustered graphs that were displayed to us using a software called Gephi. On Gephi, we can run their built-in modularity function that automatically clusters all of the

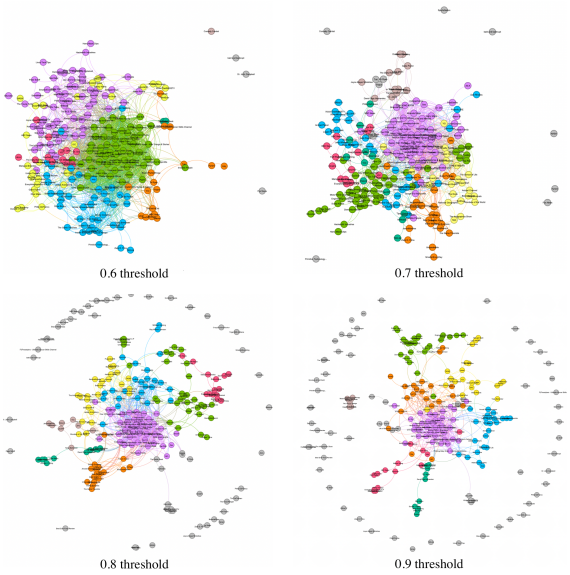

Figure 5: Graphs of 20 Keywords with 0.6, 0.7, 0.8, and 0.9 Threshold

YouTubers together and color codes them based on their category as given through the modularity. We decided it would be most important to analyze four graphs with 5, 10, 15, and 20 keywords all at the same threshold (Figure 4) as well as four graphs with the same number of keywords but all four thresholds of 0.6, 0.7, 0.8, and 0.9 (Figure 5). This way we could see if there was a common trend of how accurate our clusters were depending on the number of keywords and the threshold value.

We first analyzed the all four graphs for a threshold of 0.6; in Figure 4, it was noticeable that as the number of keywords increased, the number of youtubers in each cluster increased, and the distance between the nodes decreased. This meant that with more keywords, our system was able to better identify the overall theme of each channel and place it in its respective category. We also noticed that the graphs with more keywords started to exhibit a higher number of connections between YouTube channels. As you can see from the graph with 5 keywords, almost half of the youtubers are either in extremely tight knit clusters or not in any groups at all. However, the graph with 20 keywords only displays four YouTube channels not connected to any other nodes, a significant improvement from the five keyword graph. All other channels are also much closer to each other and have hundreds of connections running through them. We determined the main reason for this is because with a higher number of keywords, there is a higher probabil-

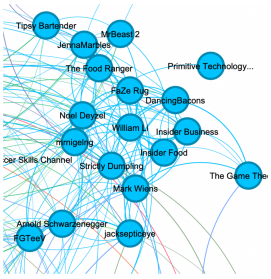

Figure 6.1 Blue Cluster

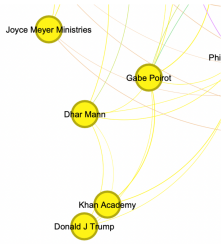

Figure 6.2 Yellow Cluster

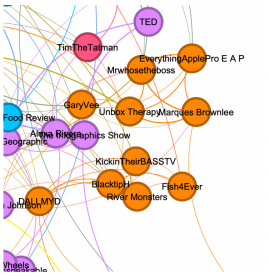

Figure 6.3 Orange Cluster

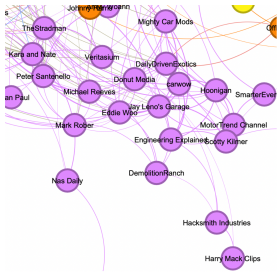

Figure 6.4 Pink Cluster

Figure 6: 4 Clusters Selected from the Graph with 20 Keywords and 0.6 Threshold

ity that YouTube channels contain identical/similar keywords. When we only extract five keywords, there is not enough information to create accurate clusters, which resulted in many YouTube channels not being a part of any group but rather stayed on the outside of the graph without any connections. Thus, the trend displayed among these four graphs is that as the number of keywords increased (from 5 up to 20) the more accurate our clustering system became.

To analyze the role threshold plays in our system, four graphs with 0.6, 0.7, 0.8, and 0.9 threshold and keyword number 20 were evaluated. As we can see in Figure 5, clusters are more well separated from each other in the graph with high threshold than those with low thresholds, which was also indicated by the trend of the modularity score when the keyword number is the same in 4.2. On the other hand, more YouTubers engage in the cluster in the graph with lower threshold than those with higher thresholds. We found out that high threshold can also result in more YouTubers not being connected to any other YouTubers because all of the cosine similarity scores are below threshold, resulting in a weighted similarity score of 0. Also, a high threshold can also result in more well-defined clusters, but the accuracy of the clustering might not be high as discussed in 4.2.

As mentioned in 4.2, the graph with 20 keywords and 0.6 threshold has the highest accuracy. Therefore, a qualitative analysis is conducted on this graph to identify the topics of a few clusters to see how relevant is the topic of the cluster to each YouTuber, and how distinct are the topics of two different clusters. The gravity of the graph was lowered so we can better read and distinguish between channels. Thus, these clusters look a little different than the graphs above.

Figure 6 includes the 6 clusters we found that can best represent the graph. Figure 6.1 is defined as a food cluster, as most of these channels pertain to food reviews and meal preparing. Still, there are a few outliers that do not belong to the food category like FaZe Rug, jacksepticeye, and Insider Business. However, as we examined, the top videos of these channels are relevant to food in some way. For example, FaZe Rug is a variety entertainment channel, but some of his top videos are about food challenges. Figure 6.2 is an example of a "vague" cluster because it is hard to identify a topic. Joyce Meyer Ministries and Gabe Poirot are religion channels; Donald J Trump is a politics channel; Khan Academy is an education channel; Dhar Mann is a variety entertainment channel. The orange cluster in Figure 6.3 is also a vague cluster, but two topics can be identified: technology and variety entertainment (mostly about fishing), so if this cluster was splitted into two clusters, a more accurate result could have been obtained. For example, Unbox Therapy's entire channel revolves around unboxing new technology items which is similar to Marques Brownlee who gives reviews on tech-related items. Figure 6.4 is the vehicle/car cluster as well as our Engineering cluster. Many of these channels talk about cars or do projects related to engineering and motor vehicles. For example, Michael Reeves is a channel that builds projects and many of his popular videos are when he creates tiny robotic vehicles.

To sum up, for the clusters that were easy to be labeled, most channels within the clusters are relevant to that specific category, and the topics of each cluster are distinct from each other. On the other hand, the vague clusters cannot represent certain topics, and some of them should be further splitted into more subclusters. Within a cluster, there might be some outlier channels that make a few videos that are relevant to the topic of the cluster, but in general the channels are not just about the topic. One possible way to get a better result was to scrape more data from the channel, so the overall content of a channel can be best

captured.

5 Conclusions and Future Work

We presented a basic model for YouTube channel clustering based on their content. From our analysis, we drew the conclusion that higher keyword count and lower threshold leads to lower accuracy and therefore, as the F-Score is below 0.5, our classification system still has some room to improve.

In our study, only Word2Vec was used to generate word embeddings. In future, more varieties of word embedding generating algorithms can be applied. In our system, Word2Vec can be replaced by context-independent algorithms like GloVe, so we can determine which word embedding algorithm can achieve a higher accuracy. However, for context-sensitive embedding algorithms like BERT, it will be challenging to make the change. Therefore, more modifications to the system need to be made to enable the use of context-sensitive embedding algorithms.

Due to the high runtime of our system and time constraint, we were not able to find a breakpoint for the threshold that after this breakpoint, as the threshold decreases, the f-score goes down, or testing our system with more thresholds to prove that a lower threshold always results in a higher F-score. Also, we were also not able to find a breakpoint or a trend for keyword numbers. After implementing a more optimized algorithm, our system can be run faster and thus allow us to try more combinations of thresholds and keyword numbers.

References

Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. TopicRank: Graph-based topic ranking for keyphrase extraction. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 543–551, Nagoya, Japan. Asian Federation of Natural Language Processing.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1):107–117. Proceedings of the Seventh International World Wide Web Conference.

Aaron Clauset, M. E. J. Newman, and Cristopher Moore. 2004. Finding community structure in very large networks. *Phys. Rev. E*, 70:066111.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Eirini Papagiannopoulou and Grigorios Tsoumakas. 2019. A review of keyphrase extraction. *CoRR*, abs/1905.05044.

Linhan Zhang, Qian Chen, Wen Wang, Chong Deng, Shiliang Zhang, Bing Li, Wei Wang, and Xin Cao. 2023. Mderank: A masked document embedding rank approach for unsupervised keyphrase extraction.

A Github Repository Link

github.com/kaiserarg/NLP-Cluster-Categorization