Pattern Recognition in Daily Top Trending YouTube Videos

Team Members:

Nathaniel Haddad haddad.na@husky.neu.edu

Winston Moh Tangongho mohtangongho.w@husky.neu.edu

Objective:

Our objective for this project is to provide YouTube content creators with a detailed trend analysis and probabilistic interpretation of what makes a top trending video on the platform in order to maximize the reach and quality of the creators' content. In this report, we will discuss our initial exploratory data analysis (EDA) of the dataset, and three experiments using the metadata of daily top trending videos on YouTube. These experiments include learning and evaluating word embeddings using word2vec, two semi-supervised learning classification sub-experiments using natural language processing, and analysis of video thumbnails using unsupervised machine learning and computer vision. Our goal is to provide insight into what daily top trending YouTube videos have in common, dig into the details of what types of attributes have the greatest impact making predictions about different attributes, and discover new relationships in the data about content.

Background:

YouTube is the largest video hosting and sharing service in the world. According to a recent study, as many as five-hundred hours of video content is uploaded to YouTube every minute[1]. First launched in 2005, the San Bruno, CA company has grown a community of over 1.8 billion users and as of August 23, 2019, ranks in second place as the most visited website, just behind search engine Google, of which YouTube is a subsidiary[2]. YouTube enables users to share, view, upload, and publish video content. Users can subscribe to a video's publisher, comment on videos, like, and dislike videos. Video content includes a variety of mediums: music videos, toy reviews, do-it-yourself tutorials, full-length motion pictures, television shows, etc.. Everyday, YouTube publishes a two-hundred item daily top trending videos list, from which the dataset used as the foundation of our research was created[3]. This list may contain many of

[1] "More Than 500 Hours Of Content Are Now Being Uploaded ...." 7 May. 2019, https://www.tubefilter.com/2019/05/07/number-hours-video-uploaded-to-youtube-per-minute/. Accessed 9 Dec. 2019.
[2] "List of most popular websites - Wikipedia." https://en.wikipedia.org/wiki/List_of_most_popular_websites. Accessed 9 Dec. 2019.
[3] "YouTube - Wikipedia." https://en.wikipedia.org/wiki/YouTube. Accessed 9 Dec. 2019.

the same entries day to day, new and old publications, as well as any of the variety of mediums published on the platform. Limitations to this list exist: YouTube does not publish videos in violation of country specific copyright laws, pornography, videos promoting violence against others, and videos with age restrictions in the daily top trending list.

For two of our experiments, we used natural language processing (NLP) techniques. It is important to provide background on these techniques. NLP is a subfield of computer science focused on machine learning using natural language in the form of text or speech. NLP allows computers to understand and process human language and is one of the earliest sub-fields in artificial intelligence. Word embedding is a technique commonly applied to NLP problems that maps words into vectors of real numbers. word2vec is one such group of models that creates word embeddings. word2vec are two-layer shallow neural networks developed by Google in 2013. word2vec neural networks are used primarily to produce continuous bag of words or skip gram representations of words. word2vec has been found to be capable of preserving high levels of semantic and syntactic relationships amongst documents.

Computer vision and unsupervised learning are interesting areas of research with useful applications such as facial recognition and image search engines. Our research into the subject gave us insight into how we might identify patterns in our datasets. We also learned about self-organizing maps: a self-organizing map performs unsupervised clustering on images when given feature vectors extracted from the images as input. In our case, the feature vectors consists of values obtained from applying filters to the images to get a response matrix. A neural network then applies filters to the image matrix to perform classification. In an attempt to perform unsupervised clustering, we used the VGG19 architecture, which is a convolutional neural network proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition"[4]. The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. The '19' in VGG here stands for the number of weight layers in the network. The network is made up of only 3x3 convolutional layers stacked on top of each other in increasing depth. Reducing the volume size is handled by max pooling. Two fully-connected layers, each with 4096 nodes are then followed by a softmax classifier. Pre-trained weights from the ImageNet dataset were used to improve the training accuracy of our model. The VGG19 model is designed to receive an image as an input and then with a high degree of accuracy will determine what subclass the image belongs to. Some subclasses include

[4] "Very Deep Convolutional Networks for Large-Scale Image ...." 10 Apr. 2015, https://arxiv.org/pdf/1409.1556. Accessed 9 Dec. 2019.

animals, sports and so on. For our use case, we performed unsupervised clustering to determine the classes for us.

There has been a lot of research into NLP, word embedding, and computer vision. However, previous published work using the YouTube daily top trending videos dataset is very limited. Professor Radivojac also suggested we read, “Will This Paper Increase Your h-index? Scientific Impact Prediction. " Author Yuxiao Dong performed similar research techniques in determining the h-score of a given scientific paper. In the case of daily top trending videos on YouTube, h-score is comparable to the ranking of each video relative to other daily top trending videos. Dong analyzed the effects of features on the h-score of millions of scientific papers, while we attempted to use our features to examine the effect they have on the rankings. We had hoped to explore our dataset in a similar manner, but did not learn enough about the dataset to make conclusions here.

Our work is particularly interesting because no one has published anything regarding experiments with word2vec, classification using NLP, and computer vision with YouTube daily top trending videos. In our EDA, we were excited to learn that the dataset contained every type of attribute we learned in class (nominal, ordinal, interval, and ratio) and gave us some creative ideas about things we could do with the dataset. Given that our dataset had a lot of text attributes, we decided to do something with NLP and word embeddings. This provided us with a great opportunity to research and learn to use word2vec and NLP techniques for classification using Scikit-learn.

Finally, in regard to computer vision, we thought it would be fun to explore a topic we did not learn about in class (computer vision, advanced clustering techniques, feature extraction, filtering, self organizing maps). Since both of us are new to machine learning, it was a great learning experience that we lucky to have. While it is great to learn new things, we also provided key insight into the thumbnail images of daily top trending videos images.

Methods:

We used a dataset available on Kaggle as we would be unable to compile our own list over a substantial amount of time. The dataset is a record dataset that contains the metadata of daily top trending videos on YouTube for several countries. We focused on the dataset for the United States, which contains 40,949 examples, of which 6,455 are unique. The reason for this drastic dropoff in videos after removing duplicates is that videos may be ranked multiple days much like the Billboard Top 40. There are sixteen

initial attributes in the dataset. As we mentioned before, the dataset contains every attribute type (nominal, ordinal, interval, and ratio). The dataset does contain outliers, but these examples are important to variations in the dataset. There was also a decent amount of missing and incomplete data. For example, YouTube does not require users to create descriptions of tags for their videos. This dataset does not contain video content, but does contain thumbnail images of each video as they would appear on the YouTube daily top trending webpage. All numerical attributes were normalized using Scikit-learn's preprocessing normalization function, MinMaxScaler. We chose MinMaxScaler because it preserves outliers as minimum and maximum values. Videos with high numbers of views are part of the dataset and are representative of trends in the dataset - we wanted to ensure we captured this.

word2vec Word Embedding:

The purpose of our word2vec experiment was to get a numerical measure of how alike document data attributes are. As this report detailed above, the daily top trending YouTube video dataset contains several text based attributes. Each example in the dataset contains a title, description, channel title, and tags (these are used for SEO purposes). We used a lot of the same feature extraction and construction methods that we learned from EDA.

After all of our text was preprocessed, we combined it into a single Pandas DataFrame made up of titles, tags, channels, descriptions, and categories. Then, we performed several sub-experiments to compare the linear relationships between each of the learned attribute examples using a proprietary algorithm we created. The algorithm, called get_similarities is a dynamic programming algorithm that averages the cosine distances between objects in two documents using pre-trained word2vec models. We then saved these values in new attribute columns for each of the word2vec models we used: a continuous bag of words model (CBOW) and skip gram model (SG). Attributes values of zero were replaced, again with the mean of the attribute. We chose this method because the mean represents the most common words each attribute shares with another.

Following the word embedding sub-experiments above, we used the new attributes learned from the word embeddings to evaluate their effect on regression. We used a two hidden layer neural network from the TensorFlow package as our model. We tested the model with and without the new attributes. Results for this sub-experiment can be found in our Results section of this report.

Classification:

For our NLP classification experiment, we cleaned the data using the same methods described above in the EDA methodology section. We also removed duplicate data because there was a high incidence of the testing set receiving examples it had already seen before, as we noted in our preliminary analysis of the dataset. For the first classification sub-experiment, we used the category attribute as the training and testing labels. For the second classification sub-experiment, we used the new_year attribute. This attribute was created by aggregating labels extracted from the initial attributes. Given our training and testing sets, we decided to use feature subset selection using the filter approach to select features to be passed to our classification models. To process text, we used a combination of two Scikit-learn TfidfVectorizers (one for words and another for characters). Term frequency–inverse document frequency (TF-IDF) is used to statistically evaluate and rank words in a raw text documents as to each words importance to the overall meaning of the document as a whole[5]. Both TfidfVectorizers were combined using Scikit-learn's FeatureUnion method. We then created a strawman classifier as our base model for both of our classification sub-experiments. We used these strawman models to create baseline metrics to which we compared different implementations of machine learning models to. The results of these sub-experiments are detailed in the Results section of this report.

Thumbnail Analysis with Computer Vision

In this second part of the project, we analyzed the dataset and scraped all video thumbnails for analysis. The goal was to use feature extraction techniques on the images, perform unsupervised clustering, and generate good hypotheses about the videos and why they were trending videos during that period of time. Each video had a link to a server which contained their respective thumbnail and we used some python code to scrape the sites and store the images in a folder. We made sure the images could be opened and that they were all jpg files. Something we realized later was that a lot of the videos had trended multiple times and so there was some duplicate images.

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \exp\left(i\left(2\pi\frac{x'}{\lambda} + \psi\right)\right)$$

Figure: Gabor filter formula to be applied to each image for feature extraction

Also, we noticed that some of the videos had no thumbnails and had no content in the image thereby giving no relevant information. There were two main tasks in this section- 1) Making sure all images were valid - that they could be opened with opencv and 2) Removing all bad thumbnail images and

[5] "tf–idf - Wikipedia." https://en.wikipedia.org/wiki/Tf%E2%80%93idf. Accessed 9 Dec. 2019.

duplicate images. To make sure the images were valid, we used opencv to open every image we had downloaded and if opencv returned a 'None' flag while trying to open the file, we removed that file from our folder. Finally, to remove the bad images from the dataset, we used one of the bad images as a reference for checking all other images. We opened every image in our folder and checked the binary content to see if the two images matched. If they did, we removed the bad file from our folder. After doing all this processing, we reduced our number of images from 6455 to 6082.

The bulk of this experiment from this section came from where we performed feature extraction from the images, performed dimensionality reduction and applied clustering to the images.

For Feature extraction, we performed both Color-based and Texture-based feature extraction methods and used these feature vectors to perform unsupervised clustering. The feature vector in the color-based extraction method had just 3 objects representing the mean RGB pixel values for each image. Meanwhile in texture-based extraction, we applied Gabor filters on each image.

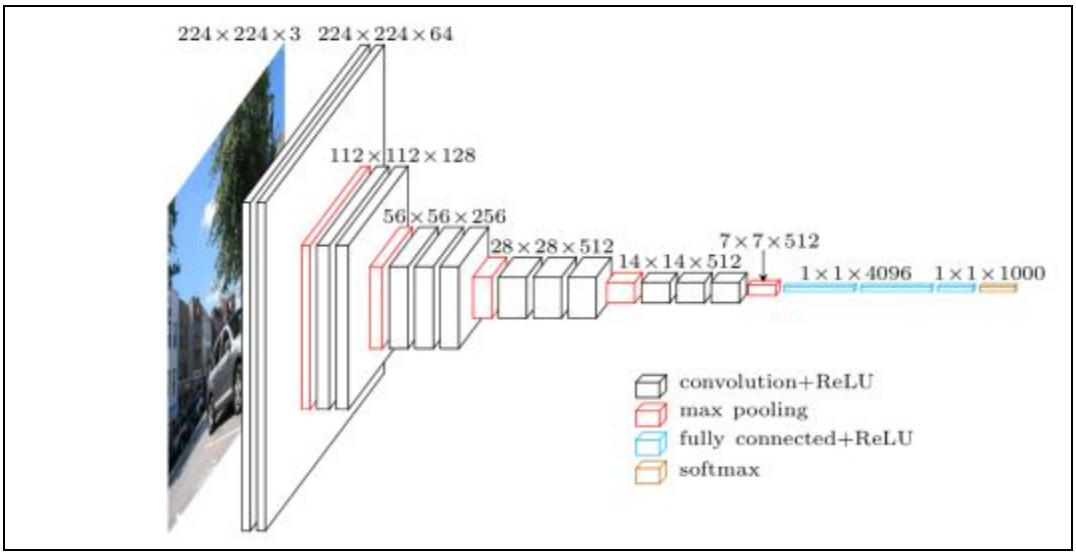
A Gabor filter[6] is a linear filter which allows a specific range of frequencies and rejects others. When applied to an image, it gives the highest response at edges and at points where texture changes. We convolved each image with 40 gabor filters at 5 different scales and 8 different orientations and obtained 40 response matrices for each image. To reduce the dimensions, we used 2 features from each convolved image - the local energy of the image (sum of the squared values of each element in the response matrix) and mean amplitude of the image (sum of the absolute value of each element of the response matrix). We appended these values to form 1 feature vector with a size of 1x80 for each image. These two features were then supplied to our clustering algorithm for data analysis. We performed K-means clustering on these two sets of feature vectors with different number of clusters and analyzed the best clustering results using the Silhouette coefficient.

To perform unsupervised learning on a large number of feature vectors from the images, we read up on self-organizing maps (SOMs) which is an algorithm that looks for similarity in input data. An SOM[7] is an Artificial Neural Network that is trained from unsupervised learning which performs dimensionality reduction on the input data and differs from other ANNs in that it performs competitive learning in which nodes fight for a right to respond to a subset of the input data unlike other ANNs which perform error-correction learning using back-propagation with gradient descent. We interpreted our results by using a U-matrix where each U-matrix value of a particular node is the average distance between the node's weight vector and that of its closest neighbors. To design our SOM, we used the Keras API to

[6] "Gabor filter - Wikipedia." https://en.wikipedia.org/wiki/Gabor_filter. Accessed 9 Dec. 2019.
[7] "Self-organizing map - Wikipedia." https://en.wikipedia.org/wiki/Self-organizing_map. Accessed 9 Dec. 2019.

create our neural network. We used the VGG19 architecture to train our Self-organizing map using the 'neupy' python API. A visualization of the architecture is shown below:

A visualization of the VGG architecture[8]



To train our model, we passed it preprocessed values of the images by subtracting the mean image values (calculated over the entire ImageNet training set). We used a sample of 1000 images in this scenario so we could gain meaningful insights. The images were first resized into the 244x244 size that the ANN required. An extra dimension was added to each image, values preprocessed and then later on they were added to an array which stored all the different images values. A small sample of thumbnail images resized to 244x244 pixels is shown below. After that, we used the entire dataset which reduced to a size of 5385 after some images couldn't be loaded correctly.

Figure. Sample of Thumbnail images

The images were then propagated to the network in batches of 16 and then trained on the VGG19 Network. Some important parameters in the SOM Network are shown below.

[8] *"VGG in TensorFlow · Davi Frossard - University of Toronto." 17 Jun. 2016, https://www.cs.toronto.edu/~frossard/post/vgg16/. Accessed 9 Dec. 2019.*

Table showing parameters for SOM Network for different number of images

N_inputs	features_grid	distance	weight	learning_radius	reduce_radius_after	std	reduce_std_after	step(learning rate)	reduce_step_after	Training epochs	Batch_size
(1000, 4096)	(20,20)	'cos'	'sample_from_data'	5	5	0.1	5	0.001	5	100	16
(5385, 4096)	(30,30)	'cos'	'sample_from_data'	5	5	0.1	5	0.001	5	10	16

No prediction was performed after training. Instead the images were resized and displayed in a 20x20 grid for 1000 images and 30x30 grid for 5385 images and the clusters were observed. The images were preprocessed to their original RGB values by reversing the preprocessing step we did earlier.

Evaluation Strategy:

For all models, we visualized our results using matplotlib and seaborn. We evaluated the effectiveness of our word2vec models by testing the built in cosine similarity functions of the models to themselves in order to guarantee that they worked correctly. This same cosine similarity technique was used to compare how similar words and documents were to each other. For our word2vec regression problem, we used mean absolute error and mean squared error to evaluate the effectiveness of learned word2vec attributes in a regression setting. For evaluation, if the model with the learned word embedding relationships performed better than the model without, we will treat that as success. For our classification experiments, we used the testing set precision, recall, f-score, and accuracy to evaluate our models. When evaluating, we used accuracy as the primary metric for evaluating the performance of each classier, but did consider all other metrics mentioned above. To evaluate the unsupervised learning we performed, we applied the K-means algorithm with different clustering sizes on the feature vectors and observed their silhouette coefficients. The better the silhouette coefficient, the better the clustering was. To observe the clustering performed by the Self-organizing map, we used a U-matrix of a fixed '20x20' size since we couldn't visualize the more than 5000 images. Then we visually observed the clustering results and looked for similarities in the clustered images. We also examined the classification accuracy of the model on the training set to see how good the model was.

Results:

word2vec Word Embedding: In our experiment, we used the Gensim word2vec package to extract features from the YouTube daily top trending dataset by creating word embeddings for each of of the text-based attributes and then pitted them against each other to determine cosine similarity. We learned that we could use word2vec to compare word embeddings to examine linear relationships between different documents for each of the attributes and examples in our dataset. In order to verify that every document and attribute was embedded correctly, we applied our get_similarities algorithm to compare documents. For a result, we achieved a mean cosine similarity of 0.1000 between the title attribute and itself for both of the CBOW and SG word2vec models.

For our first word2vec sub-experiment, we compared the title attribute with the tag attribute for each example in the dataset. The results showed a clear relationship between titles and tags. We achieved a mean cosine similarity between titles and tags of 0.9501 using a CBOW model and a mean cosine similarity 0.8953 using a SG model.

Evaluating Relationships of Video Titles and Tags

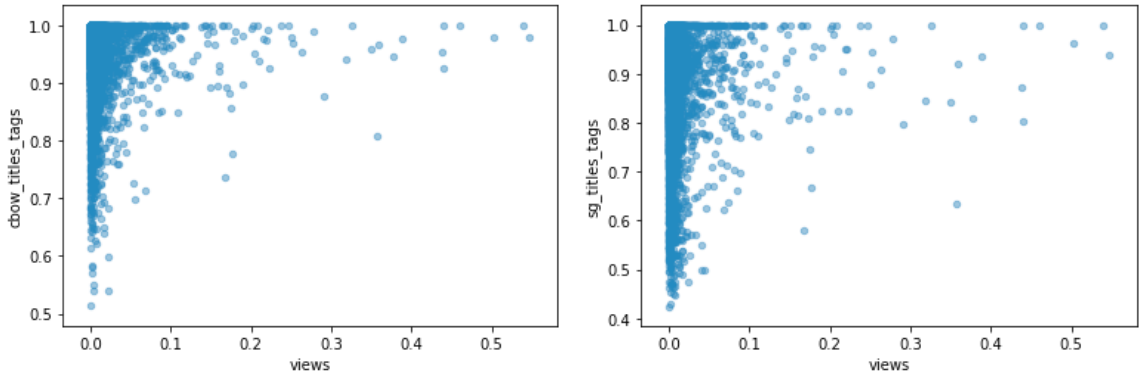


Figure: word2vec linear relationships between titles and tags attributes. (Left) CBOW (Right) SG

These results are due to that fact that many tags include words that are similar to the titles. word2vec models learn word embeddings incredibly well, and therefore, will assign similar values for words and phrases that have a stronger linear relationship. It is also worthy to note that some of these tags contain the same words as the title. This is to be expected as well, due to the fact that many video titles contain the words most commonly used to search for them.

For a second sub-experiment, we used word embeddings to achieve a mean cosine similarity between titles and descriptions of 0.9533 using a CBOW model and a mean cosine similarity of 0.8998 using an SG model. Again, this is means words in the title often appear in the description. We saw a significant

dropoff in linear relation between title and channel title attributes when compared using word2vec. Given that channel titles are usually only a few words long, this is not surprising. We found mean cosine similarity between titles and channels of 0.7713 using a CBOW model. More importantly, we saw a significant drop in mean cosine similarity between titles and channels for a SG model: 0.6689. The results of our last experiment showed no relationship between titles and categories. Our CBOW model gave a mean cosine distance between title and category attributes of 0.6135. Even more interesting, the SG model mean cosine similarity between titles and categories attributes was extremely low 0.3729. Similar to what we reported in the previous experiment, we learned that because category values are only a few words long and do not contain channel titles, they perform even poorer than the previous experiment of comparing titles with channel titles.

Evaluating Relationships of Video Titles and Categories

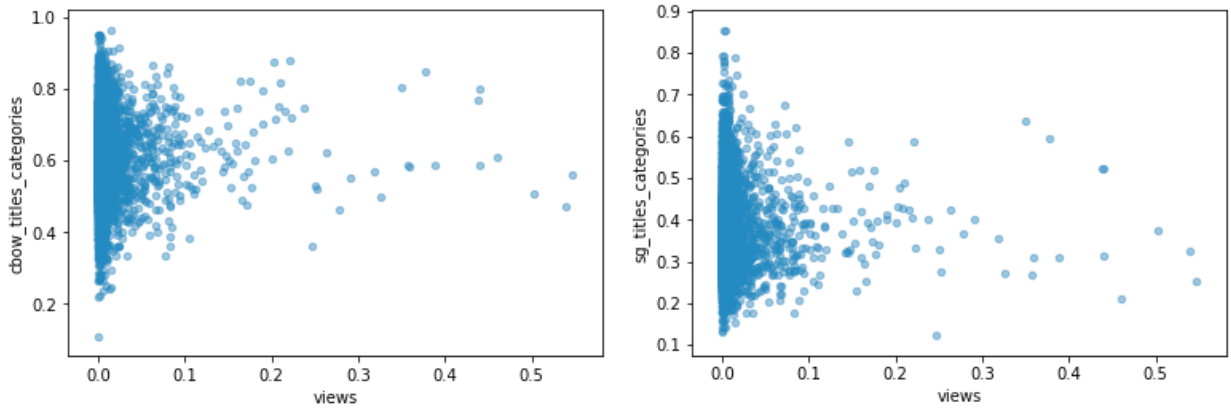


Figure: word2vec linear relationships between title and category attributes. (Left) CBOW (Right) SG

We were able to achieve some interesting results from regression using the features extracted from the data using the word2vec models. It is important to note our purpose here: we are simply trying to evaluate the usefulness of these learned relationships from the sub-experiments on regression. Our goal here is not to build the best regressor possible for the given data. Using a feed-forward neural network with two hidden layers of 150 neurons each, we attempted to predict the number of views using the values learned from our word2vec sub-experiments and without the values learned from our word2vec sub-experiments. We tested two regressors: one with the values learned from word2vec, one without:

Performance Metrics for Regression Using word2vec

Model	**Test Loss**	**Test MAE**	**Test MSE**
w/o word2vec	5.8617e-05	0.0038	5.8617e-05
word2vec	4.1845e-05	0.0037	4.1845e-05

From our results, we are able to determine that the word2vec regression model performed much better than compared to the strawman feed-forward neural network. The improvement is significant and demonstrates that word2vec is able to learn word embeddings that can then be used to make predictions about unseen data. We visualized the performance of these models to help support our findings.

Regression with Learned word2vec Values

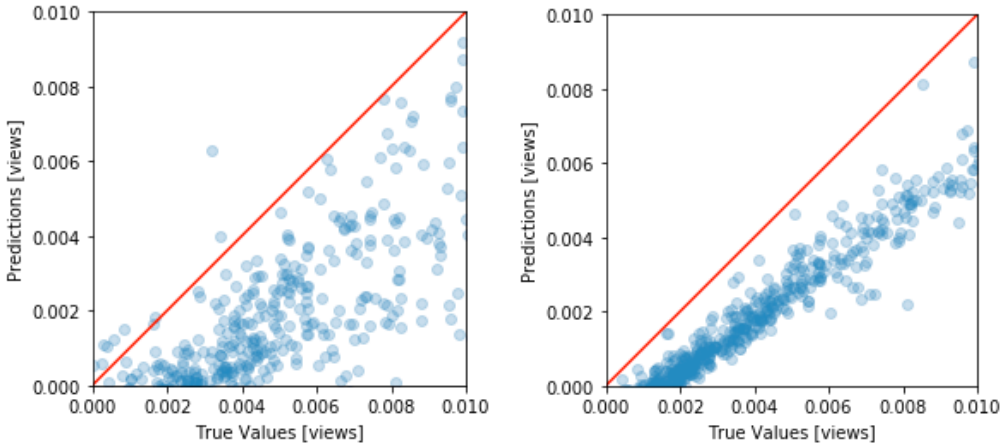


Figure: (left) ANN without word2vec values. (Right) ANN with word2vec values. Notice the density of points at the origin. This is a result of the majority of videos in the dataset have a similar number of views when they appear on the list for the first time.

Semi-Supervised Classification:

The following section includes the results for two classification sub-experiments. In the first, we classified video category using NLP and text based attributes. Getting started, we knew right away that in order to attain decent accuracy, we needed to use a model that could handle our initial sixteen valued category label. For the strawman classifier, we used Scikit-learn's MultinomialNB model. The MultinomialNB model contains built-in smoothing, which was extremely helpful in handling the datasets unequal distribution of label values. We also included the GridSearchCV method in this strawman so that our base model was optimal. In the following model, we used LogisticRegressionCV, using a multinomial distribution setting and solver set to Newton-conjugate gradient to allow for convergence. For this problem, we did not include GridSearchCV, as we hand selected and tested the parameters of the function, which already includes 10-fold cross validation. At this point, we knew we had to do something about the sixteen classes in the category label. Although we attained a decent classifier, reducing the number of labels would greatly increase stability. To put it simply, there are not enough examples for each class for our models to perform with a high level of accuracy, and without count matrices, we need to devise alternate solutions. We solved this problem by combining the category label classes into two classes such that a binary classification problem could be performed. Our two classes, Entertainment and Informational, were created from the sixteen previous classes. While we aggregated these classes using

human analysis, we believe a better way of creating two new classes would be to use the word2vec model to label examples based off of cosine similarities. After creating our new labels, we tested BernoulliNB as an alternative to the strawman model and LogisticRegressionCV. We achieved the following:

Performance Metrics for all Classifying Category models

Model	Test Precision	Test Recall	Test F-Score	Test Accuracy
MultinomialNB	0.79363	0.75965	0.75197	0.75965
LogisticRegressionCV	0.80415	0.80693	0.80102	0.80693
$\text{BernoulliNB}_{[1]}$	0.90533	0.90544	0.90527	0.90544
$\text{LogisticRegressionCV}_{[1]}$	0.91006	0.91017	0.91003	0.91017

Figure: [1] model after labels aggregated into a binomial distribution.

We were unable to achieve great results for our second classifier: classifying videos by publishing year. However, we were able to get high enough metrics to verify that it is possible. For the publishing year attribute, we learned that more than half of the videos are from 2018, while the remaining number of videos are from 2017, with a few outliers (videos that were published in earlier years, but became a trending video at a later date). We decided to combine pre-2018 videos to create two labels: 2018, and pre-2018 in order to create a binary classification problem. We followed the same procedure described above in our methodologies section and above in the Classifying Category Results section. For our strawman model, we used BernoulliNB, then tested LogisticRegressionCV, and finally, a MLPClassifier.

Performance Metrics for Classification Publishing Year

Model	Test Precision	Test Recall	Test F-Score	Test Accuracy
BernoulliNB	0.71486	0.72656	0.71502	0.72656
LogisticRegressionCV	0.76914	0.77541	0.76616	0.77541
MLPClassifier	0.74145	0.74783	0.72681	0.74783

While we did not accomplish as much as we had hoped to with this sub-experiment, if we had more time we believe that with other types of word embeddings and feature extraction methods, we might be able to attain better results for our Classify Publishing Year sub-experiment.

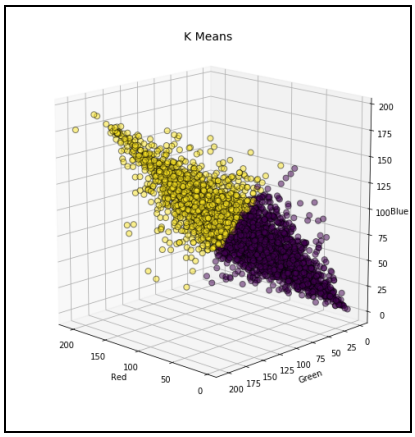
Unsupervised Learning on Thumbnail Images

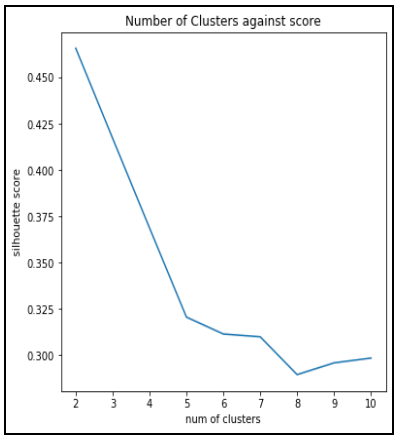
First, we performed K-means clustering on the images' feature vectors to see how they were related and obtained the results in figure (a). Where, the 3 axes represented Red, Blue and Green pixel values. We then created 40 Gabor filters with different orientations and at different scales using the openCV

getGaborKernel function and convolved the images with filters using the filter2D function. This filters were used to create the second feature matrix which was also used in K-means clustering.

Next, we evaluated our K-means clustering results by evaluating their silhouette coefficient against the number of clusters for both color-based (RGB) and texture-based(Gabor-flitered images) features and obtained the results in figures(b) and (c) respectively.

K-Means Clustering of Color and Texture Features in Thumbnail Images





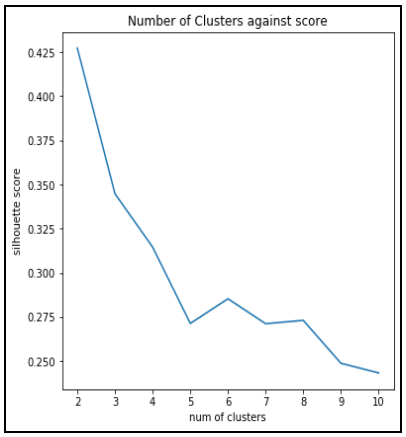


Figure (a). K-means with 2 clusters. fig.(b) fig.(c)

The big observation here was that the better score was obtained from using 2 clusters and that proved that the feature vectors have very high densities and so K-means wasn't a good way to find similarities between images even after using Gabor filters and getting feature vectors from them. In response, we trained an ANN to learn the similarities between the images vectors and place them in a U-Matrix. The ANN used the VGG19 architecture previously discussed with pre-trained weights to correctly classify the pictures. The U-matrix produced a better representation of the images and to visualize the results, we used 1000 images for a 20x20 grid and for all the images we used a 30x30 grid. We trained our SOM on the output from the VGG19 network and obtained the following accuracy results after 100 epochs.

Training Error on 1000 images on VGG network

```
#98 : [3 sec] train: 2.017234
#99 : [2 sec] train: 2.017234
#100 : [3 sec] train: 2.017233
```

Training error on 5385 images on VGG Network

```
#8 : [43 sec] train: 2.011671
#9 : [43 sec] train: 2.011666
#10 : [37 sec] train: 2.011546
```

The error rate of 2.0172% was good for 1000 images and was even better at 2.011% for all the images. Therefore predicted the image classes with a good accuracy and with more images to classify the accuracy got better. We trained the SOM for just 10 epochs for the entire dataset because it took a very long time to train but there wasn't much change even after 10 epochs. As you can see it took about 40 seconds to train on one epoch.

Finally, to visualize the clustered images, we created our U-Matrix and analysed the output from our Self-Organizing map. The U-Matrix on a sample set of 1000 and 5385 images is shown below.

U-Matrix after training SOM with 1000 thumbnail images

Analysing the U-matrix for 1000 images, some deductions can be made. The SOM grouped the music videos in one corner, men with firsts in another corner, images with Will smith were grouped in the same area, facial make-up videos in another area and food images were clustered towards the top right of the matrix.

U-Matrix after training SOM with 5385 thumbnail images

Analysing the U-matrix for the 5385 images, some deductions can be made. The picture shown here has been resized to fit so it is a little harder to see the clusters. The SOM did a good job of grouping Talk shows together just as before, basketball player images were also well grouped, CNN News images were also grouped together. Not all the images were in the matrix as the matrix only displayed those which fitted perfectly according to the distance metrics for clustering.

Taking a deeper dive, we see that images with Talk shows hosts such as Conan, Jimmy Kimmel had a high occurrence rate and hence those videos went viral more than others towards the top right corner.

Also, music videos with the channel ‘vevo’ had a large clustering group towards the bottom and middle part of the matrix. Hence, music videos from the Channel ‘vevo’ had a high chance of going viral.

Conclusions:

word2vec Word Embedding and Regression

From this experiment, we can conclude that are significant relationships between the word2vec word embeddings of daily top trending YouTube videos. Our research demonstrates that a majority of daily top trending videos contain descriptions, tags, and titles that are similar to each other. For attributes like title and tag, similarities exist simply because of SEO: users are often searching for the words in the title of a video in order to find it. With attributes like channel title and category, relationships can be harder to identify: word embeddings are more difficult to learn with fewer words per document; channel titles are not often included in video titles; category names also do not appear often in titles. Nonetheless, we still learned meaningful information about our dataset and about daily top trending videos in general that can be applied to future research about the topic.

Classification

We can conclude that it is possible to successfully classify YouTube videos as entertainment or informational. We were also able to classify videos according to their sixteen classes with category as the label. However, there are not enough examples in our dataset from each of the sixteen classes. While we are able to classify videos as informational or entertainment using NLP, with more examples we could make our multi-class classifier work even better.

We also classified videos by year and achieved little success. We hypothesis that with more time, we would have experimented with samples of the n-largest videos from each year, removing examples published in outlier months such as January and December from the dataset, or including different features using filtering feature selection or other feature extraction techniques, we could have gotten better performance out of our strawman and other models. Given that we worked exclusively with text-based attributes, we were not able to achieve a high enough score that would provide us with enough confidence to make assertions about the dataset.

Computer Vision

We can conclude that the images from the dataset after clustering them by thumbnail similarities showed that that some images with celebrities or popular faces generated a lot of views. So having a popular face in your thumbnail definitely drives up the number of views. Using the entire dataset and observing the 30x30 matrix, we discovered that channels such as CNN, CONAN, Jimmy Kimmel Live, Vevo music videos and movie trailers had the most appearances in the Top trending lists. This in our opinion implies that very popular channels with celebrities more often than not make it to the Top Trending lists. For smaller channels, the task is harder but not impossible as we observed that smaller channels in the matrix but with just 1 or 2 appearances hence the odds of trending are a lot lower.

Major drawback of using the VGG19 Network was the amount of time it took to train the network. With our large dataset, it took up to an hour to train the network. The use of 19 weight layers also improved the training rate and improved the classification accuracy. A future improvement will be to see whether the clustering from the SOM corresponds to the clustering we would have gotten if we used the video categories as classes. Another improvement would have been to display the number of views in the U-matrix and from there we could see if any classes had more views than others and could be the Key to getting more views on Youtube and be ranked as a top Trending video. There wasn't enough time to explore these ideas but that would've given us some more insights. Nevertheless, Visualizing the clusters in our U-Matrix was quite informative and showed us how unsupervised clustering could be done on images.

All implementation details can be seen in the '.ipynb' files attached to file.

Individual Tasks:

Nathaniel Summary

For my portion of the project, I completed the EDA experiment, word2vec experiment, and classification experiment. These experiments correspond to the cs6140_project_EDA.ipynb, cs6140_project_Word2Vec.ipynb, and cs6140_project_Classification.ipynb Jupyter Notebooks. For each of the experiments listed above, I preprocessed and cleaned the datasets, researched and implemented methods and models, and wrote about my findings in this report. In terms of specifics, I performed EDA of the dataset, used word2vec models to create word embeddings for all of the text-based attributes in the dataset, used the word embeddings to compare cosine similarities between different text-based attributes, and then used these in TensorFlow regression models. I implemented two NLP classification sub-experiments with a total of seven classifiers implemented - all with some degree of success. I measured performance using the techniques discussed in this report and visualized my findings.

Winston Summary

For my part, I did some internet scraping to download the thumbnail images, deleted bad or duplicate images and performed K-means clustering using feature vectors - RGB feature vectors and Gabor filtered feature vectors. K-means clustering was done with different number of clusters and I performed some accuracy analysis of the different cluster numbers using the Silhouette coefficient. Next, I performed unsupervised clustering by using the VGG19 architecture and trained a Self-Organizing map to correctly cluster the images into classes. I downloaded pre-trained weights from the ImageNet results, used them in my Artificial Neural Network and then drew a grid known as a U-Matrix which showed the images clustered with similar images according to the image content. I had two different grids for different number of images - 1000 and 5385. Then I analyzed the accuracy results of the ANN and also talked about what improvements could be done with the Neural Network and U-Matrix to gain better insights.

References:

[0] https://www.businessinsider.com/youtube-user-statistics-2018-5

[1] https://www.tubefilter.com/2019/05/07/number-hours-video-uploaded-to-youtube-per-minute/

[2] https://en.wikipedia.org/wiki/List_of_most_popular_websites

[3] https://en.wikipedia.org/wiki/YouTube

[4] https://www.youtube.com/feed/trending

[5] https://www.cbsnews.com/news/top-10-highest-paid-youtube-stars-of-2018-forbes/

[6] https://en.wikipedia.org/wiki/Google_AdSense

[7] https://support.google.com/youtube/answer/7239739?hl=en

[8] https://www.kaggle.com/datasnaek/youtube-new

[9] https://en.wikipedia.org/wiki/Self-organizing_map

[10] https://en.wikipedia.org/wiki/Gabor_filter

[11] https://en.wikipedia.org/wiki/Tf%E2%80%93idf

[12] https://www.tensorflow.org/tutorials/keras/regression

[13] https://www.cs.toronto.edu/~frossard/post/vgg16/

[14] https://arxiv.org/pdf/1409.1556.pdf

[15] Dong, Yuxiao, et al. “Will This Paper Increase Your h-index? Scientific Impact Prediction” Johnson, Reid A., Chawla, Nitesh V., *Proc. of the 8th ACM International Conference on Web Search and Data Mining (WSDM'15)* https://arxiv.org/abs/1412.4754

[16] CS6140 Machine Learning Class Resources