Categorization of YouTube Videos

Nitesh Gupta
Computer Science
Iowa State University
nkgupta@iastate.edu

Tanmay Gore
Computer Science
Iowa State University
tgore03@iastate.edu

Abstract

Classification of web-based videos is an important task in video search and ads targeting applications. The proposed method categorizes YouTube videos in different genres like Comedy, Horror, Romance, Sports, Technology in a supervised manner. It maps the different features of a video to TF-IDF vectors in a sparse zero matrix. Dimensionality reduction is performed using PCA on the obtained features. Fully Connected Feedforward Neural Networks and k-Nearest Neighbor algorithms are used for the training purpose. The method then computes a new TF-IDF vector array for the new videos and classifies it to one of the categories obtained. The results obtained from both the models are compared based on running time and classification accuracy.

Keywords: TF-IDF, PCA, YouTube, Neural Networks, k-Nearest Neighbors, Classification.

1. Introduction

Categorization of videos is an increasingly prominent area of research, rising with the increasing number of videos shared through online platforms such as YouTube. Its applications are of paramount importance to video search and website monetization through advertisements. However, the classification of videos to various category poses a great challenge. An accurate classification of videos can help to quickly suggest new videos to the user. It also provides the monetary benefit to ads targeting applications.

In this paper, we evaluate the efficacy of two different machine learning models for video categorization. This project includes a categorization based on few selected features of a video. The selected features are tags, video title, and video description.

This project explores four main ideas – TF-IDF, PCA, Neural Networks and k-Nearest Neighbor. TF-IDF stands for *term frequency-inverse document frequency*, and the TF-IDF weight is often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance of a word is proportional to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.

The following are the equations for TF and IDF respectively:

$$TF(t) = \frac{Number\ of\ times\ t\ occurs\ in\ a\ document}{Number\ of\ words\ in\ that\ document} \quad \ldots (1)$$

$$IDF(t) = log_e\left(\frac{Number\ of\ times\ t\ occurs\ in\ a\ document}{Number\ of\ words\ in\ that\ document}\right) \quad \ldots (2)$$

The TF-IDF is the product of equation (1) and (2).

The second concept that we explore is Principal Component Analysis (PCA). PCA is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. This transformation is defined in such a way that first principal component has the largest possible variance and each succeeding component has the highest possible variance under the constraint that it is orthogonal to the preceding components. A data point can have thousands of features. The running time for any machine learning algorithm depends on the number of data points in the dataset and the dimensionality of a data point. Through PCA it is possible to achieve the faster running time for the dataset by reducing the dimensions of a data point while preserving most of its value. Thus, PCA plays an important role in the situation when there is a need to perform various algorithms on a dataset to obtain the comparison results.

Artificial Neural Network (ANN) is an important tool used in machine learning. These are the brain-inspired systems that are intended to replicate the ways the humans learn. An ANN is based on a collection of connected units called artificial neurons. Each connection between neurons transmits a signal from one to another. The output of each artificial neuron is calculated by a nonlinear function of the sum of its input. These connections also have weights that adjust the learning process. Neural Networks consists of the input layer, hidden layers, and an output layer. Hidden layers play the role of transforming the input into the form desired in the output. These are excellent tools for finding computationally intensive patterns underlying the data.

k-Nearest Neighbor is an algorithm used for classification and regression in the field of pattern recognition. The input consists of k closest training examples in the feature space. In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors.

2. Methodologies

To categorize a video, we perform three major steps: (1) Extract 20000 videos from YouTube (2) Computing TF_IDF to convert the string features into equivalent numeric features. (3) Applying PCA for the dimensionality reduction.

To obtain a good classification accuracy, we train two models: (1) Train the machine learning model using Feed Forward Fully Connected Neural Networks and get the running time, test accuracy and confusion matrix. (2) Apply k-Nearest Neighbor for the classification of the data among k categories and get the running time, test accuracy and confusion matrix.

2.1. Extraction of the YouTube videos

YouTube provides two APIs for retrieving video information. The Streaming API provides methods to retrieve videos in bulks and filter them out based on location, keywords etc. The REST API provides methods to retrieve video-specific information such as video title, description, tags information etc. We use this API

to get most trending US videos and to classify them into various categories.

2.2. Converting Video information to TF-IDF vectors

Since machine learning models don't work well with string, the video information needs to be converted to the numerical representation. We first assign a score for every word in video tags. This would enable us to have a vector representation of every feature that was obtained. Once the scores were obtained, a set containing every word was made. Let's call this set "*wordSet*". Every word in *wordSet* was assigned a unique index (*index word*). Next, a sparse zeros array containing dimensions [number of videos, length of *wordSet*] was generated. We iterated through the scores for every word for each video and added them to the specific index as determined by *index word*. This was done for every video in the sample space. Thus, the array contained the TF-IDF vectors for each video.

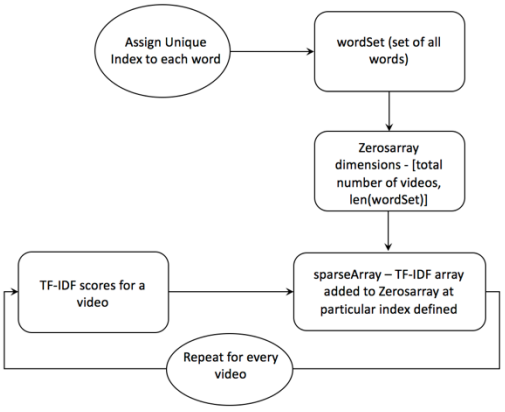

Figure 1: Block diagram illustrating clustering phase

2.3. Dimensionality Reduction for the YouTube Videos

Due to the high dimensionality of TF-IDF vectors, the machine learning algorithms would take a long time to train and predict the categories. PCA is dimensionality reduction algorithm which combines multiple dimensions into a single dimension with minimal loss of information. Thus, providing the same data with reduced dimensions.

PCA is applied to obtain a new matrix with the reduced dimensionality of each vector. This facilitated a dimension reduction from 12000 to 3000 dimensions with 95% of the information retained.

2.4. Classification Methodology

This phase of the algorithm deals with classifying the new videos to the specific categories. The classification is obtained using two methods, Feed Forward Fully Connected Neural Networks and k-Nearest Neighbors.

We use the KNN API provided by Scikit – Learn package to train the k-Nearest Neighbor model. We build the feed forward neural network using Tensorflow and Keras with 2 hidden layers and 1 input and output layer each.

We later use the dimension reduced matrix and the predefined categories obtained from YouTube API to train each of the two models. A subset of the matrix and labels are kept aside for testing while the remaining data is used for training the model. Each of the two models then predicts the categories on testing data and the result is used to calculate the classification accuracy and obtain the confusion matrix.

The results obtained from Neural Networks and k-Nearest Neighbor are compared based on their accuracy and the running time.

3. Experiments

We tested the overall accuracy of the models with different features which include Video Title, Video Description, and Tags for the video. The experiment is also performed with various values of parameters for the machine learning models.

3.1. k-Nearest Neighbor

The experiment is done with different values of k from 1 to 20 to obtain the category for a video. Also, various set of features are used for the training purpose to obtain the best classification model.

3.2. Neural Networks

Neural Network used for the training purpose is Feed Forward Full Connected Neural Network. SoftMax function is used for the output layer and "1 of c" encoding is used. The early stopping technique is used to stop the training when the optimum training accuracy is obtained. Experiments are performed using following values for different parameters:

1) Error Function: Sum of Square Error, Cross Entropy Error
2) Hidden Units:
    a. Number of Neuron in the hidden unit: 100, 500, 1000, 3000
    b. Number of hidden layers: 2, 3
    c. Type of hidden Units: ReLU, Logistic, Linear
3) Learning Rates:
    a. 0.05, 0.01, 0.1
4) Momentum Rate:
    a. 0.9, 0.7, 0.4
5) Different Features:
    a. Tags, Tag Count
    b. Description
    c. Title
6) Batch Size:
    a. 100, 200, 300, 400, 500
7) Epochs:
    a. 10, 50, 100, 200, 300
8) Optimizer:
    a. Stochastic Gradient Descent
    b. Adam

4. Results & Discussions

We found that Artificial Neural Networks (ANN) performed a better job of classifying the YouTube videos than k-Nearest Neighbor (k-NN). The best ANN model was obtained with Categorical Cross Entropy Error Function, Linear hidden layers with 3 layers and 500 neurons in each layer. However, k-NN performed more consistently when for different Hyper-Parameter values as compared to ANN. k-NN gave the highest accuracy of 67.7% on the training set while ANN gave the highest accuracy of 73.6%. We were able to achieve an accuracy of about 92 percent on our test dataset with the ANN while k-NN gives the best accuracy of around 76 percent.

We started with only one feature (tags) to obtain the training accuracy and subsequently included all the features. There was an increase in the accuracy of the k-NN model from 64.36 % to 67.3%. Also, the accuracy of ANN increased from 64.2% to 73.6%. We got the best accuracy for Tags, Description, Title, Tag Count. The dimensionality reduction with PCA decreased the runtime by a significant amount. Our reduction preserved 95% of the information and reduced the dimension from 57000 to 4000 which is a significant reduction.

Table 1: Best Parameter Values obtained for ANN

Neural Network Hyper Parameters	
Error Function	categorical_crossentropy
Hidden Units	Linear
Number of Hidden Layers	3
Neurons in each hidden unit	500
Learning Rate	0.05
Momentum Rate	0.4
Features Used	Tags, Title, Description
Accuracy	67.2 percent
Epochs	10
Batch Size	100
Optimizer	Adam

CLASS 0 : 0.5435	CLASS 7 : 0.4487
CLASS 1 : 0.1333	CLASS 8 : 0.726
CLASS 2 : 0.7963	CLASS 9 : 0.8
CLASS 3 : 0.68	CLASS 10 : 0.8817
CLASS 4 : 0.9074	CLASS 11 : 0.875
CLASS 5 : 0.5556	CLASS 12 : 0.8333
CLASS 6 : 0.4286	CLASS 13 : 0.6607
	CLASS 14 : 0.0

Table 2: Class Conditional Accuracy for ANN

[25	0	2	0	0	0	0	1	0	14	1	2	0	1	0]
[0	2	2	1	0	0	1	2	0	5	0	0	0	2	0]
[2	0	86	0	1	0	0	1	0	16	0	0	0	2	0]
[1	0	1	17	0	0	0	0	0	3	3	0	0	0	0]
[0	0	0	1	49	0	0	0	0	0	4	0	0	0	0]
[0	0	0	0	0	5	0	2	0	2	0	0	0	0	0]
[0	0	0	0	1	0	6	3	0	3	0	0	0	1	0]
[1	0	6	1	2	1	0	35	0	15	4	10	0	3	0]
[3	0	1	0	0	0	0	0	53	14	1	0	1	0	0]
[10	1	8	0	3	0	0	9	2	172	1	4	2	3	0]
[0	1	0	0	0	0	0	5	0	4	82	1	0	0	0]
[0	0	0	0	0	0	0	0	0	8	0	70	1	1	0]
[0	0	0	0	0	0	0	0	0	2	1	1	30	2	0]
[1	2	0	0	0	0	0	2	0	4	4	0	6	37	0]
[0	0	1	0	0	0	0	1	0	2	2	1	0	0	0]

Table 3: Confusion Matrix for ANN

k Nearest Neighbors	
k	5
Features Used	Tags, Title, Description, Tag Count
Accuracy	67.7 %

Table 4: Best Parameter Values obtained for k-NN

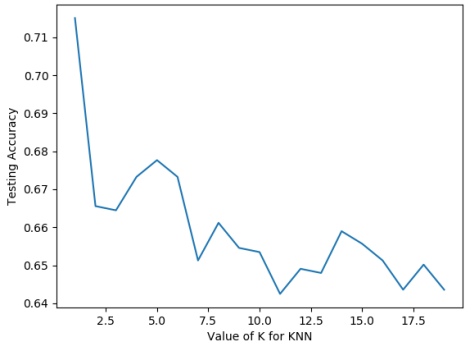

Diagram 2: k-NN performance for k Values

CLASS 0 : 0.3409	CLASS 8 : 0.4407
CLASS 1 : 0.4167	CLASS 9 : 0.7529
CLASS 2 : 0.8108	CLASS 10 : 0.694
CLASS 3 : 0.0	CLASS 11 : 0.8085
CLASS 4 : 0.75	CLASS 12 : 0.6538
CLASS 5 : 0.7538	CLASS 13 : 0.5714
CLASS 6 : 0.3333	CLASS 14 : 0.6545
CLASS 7 : 0.3636	CLASS 15 : 0.0

Table 4: Class Conditional Accuracy for k-NN

[**15**	0	4	0	0	1	1	0	2	3	16	0	1	0	1	0]
[0	**5**	0	0	0	0	0	0	0	0	4	2	0	0	1	0]
[1	0	**90**	0	1	0	0	0	3	2	10	2	1	0	1	0]
[0	0	0	**0**	0	0	0	0	0	0	0	0	0	0	1	0]
[0	1	1	0	**15**	0	0	0	2	0	1	0	0	0	0	0]
[3	0	1	0	1	**49**	0	0	3	0	4	4	0	0	0	0]
[0	1	0	0	1	0	**4**	0	1	0	2	2	1	0	0	0]
[0	0	1	0	1	1	0	**4**	0	0	3	0	0	0	1	0]
[1	1	9	0	1	1	0	1	**26**	2	9	3	2	1	2	0]
[0	0	1	0	1	2	0	0	2	**64**	13	1	1	0	0	0]
[5	0	14	0	2	4	3	3	12	10	**161**	7	9	0	2	0]
[0	0	2	0	1	4	0	0	3	1	3	**76**	0	1	3	0]
[1	0	2	0	0	2	0	0	9	0	11	1	**51**	0	1	0]
[0	0	0	0	1	0	0	0	2	1	2	1	5	**16**	0	0]
[1	0	2	0	1	0	0	0	6	0	2	1	4	2	**36**	0]
[0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	**0**]

Table 5: Confusion Matrix for k-NN

5. Conclusion

This project focused primarily on obtaining a good classification model for YouTube Videos. This project tries to obtain the best model in terms of Accuracy and Running Time by Preprocessing the data and using Neural Networks and k-Nearest Neighbors. We successfully obtained a Neural Network model that can classify the videos with a 73.6% accuracy. To improve the running time, we used PCA which helped in reducing the dimensions from 57000 to only 4000. TF-IDF not only helped to represent video description in numerical format but also helped to classify the videos based on meaningful words rather than useless words such as stop words. Thus, we were able to obtain a model with very good accuracy, consistency and running time.

6. Member Contribution

This Project team included two members and their contributions are as follows:

6.1. Tanmay Gore

Worked on extracting the data from file, separating labels from data, storing extracted data to file and converting textual features to numerical formats using TFIDF. Also worked on Neural Network model and optimized it for better accuracy. Responsible for writing the experiments and Results in this report and preparing slides for introduction, TFIDF and PCA.

6.2. Nitesh Gupta

Worked on applying PCA and Isomap on data obtained from TFIDF, building KNN model, Plotting the data and printing statistics for training and test data. Responsible for writing Abstract, Introduction, Methodology, Conclusion and references in this report and preparing slides for KNN, Neural Network and Conclusion, References.

7. References

1. http://www.tfidf.com/ n.d.
2. https://en.wikipedia.org/wiki/Artificial_neural_network
3. https://en.wikipedia.org/wiki/Principal_component_analysis
4. https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
5. https://www.kaggle.com/datasets