2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)

YouTube Video Classification based on Title and Description Text

Gurjyot Singh Kalra
Department of Computer Science and Engineering
HMRITM
Delhi, India
singh.gurjyot08@gmail.com

Ramandeep Singh Kathuria
Department of Information Technology
HMRITM
Delhi, India
singh.kathuria@gmail.com

Amit Kumar
Department of Computer Science and Engineering
AIACTR
Delhi, India
amitkr6002@gmail.com

***Abstract*— YouTube has a library of millions if not billions of videos and keeping a track of the types of videos for effective retrieval and use can be quite difficult. YouTube videos can be classified into different classes based on the title and descriptions of the videos. To classify so many videos, an effective scalable algorithm is required. This can be achieved by using a Random Forest Classifier along with Natural Language Processing techniques like Bag of Words, Word Stemming etc. This paper also discusses method to scrape YouTube videos using packages like selenium, requests and Beautiful Soup for videos and their metadata. At the end we discuss various evaluation metrics for Random Forest Classifiers.**

***Keywords*— Random Forest, Text Categorization, Stemmer, BeautifulSoup, Selenium.**

I. Introduction

YouTube videos can be classified using numerous techniques like classifying videos based on their content, comments on videos, and many more metrics. But these methods can be complex and time consuming[1]. Our approach to classifying videos on YouTube lies in Text Classification, more specifically title and description classification. We used the titles and descriptions of several YouTube videos to classify them into 6 categories:

- Travel
- Science and Technology
- Food
- Manufacturing
- History
- Art and Music

Using BeautifulSoup, requests and Selenium python libraries, we scraped YouTube to get titles and descriptions of videos belonging to each of the 6 categories for our training and testing data. We managed to scrape data for around 6000 videos. The collected data needed to be cleaned before feeding it to the Random Forest Classifier. We used NLP techniques to remove stopwords, Lebel Encode the categories and used PorterStemmer to make sense of words in sentences. Stemmer is used to strip the word and reduce it to its stem form to get ahold of the meaning of the word rather than variations in the word[2]. Then we made Bag of Words of 1500 words from titles and descriptions to be fed into the classifier using CountVectorizer.

Figure 1 explains the life cycle of our project. First Data Collection is done using requests, selenium and BeautifulSoup. Then processing of the data using NLP techniques like removing stopwords, stemming and making the bag-of-words. Model is trained using this data and finally evaluation of the model takes place using various metrics.

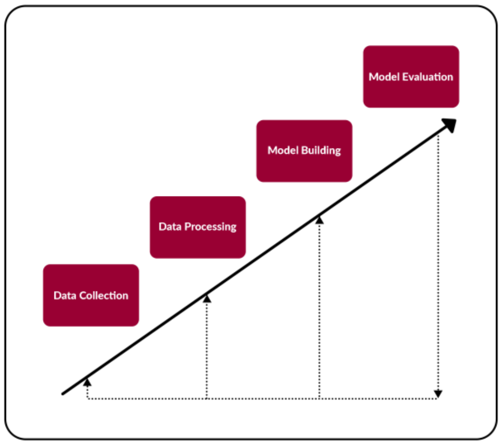

Figure 1. Project Life Cycle

II. Data Collection

Scraping video data from YouTube is not a straight-forward task. In order to get titles and descriptions of the videos, first we need to get the IDs' of the videos[3]. The Video IDs' can be retrieved by the search results page. The hurdle we faced was that YouTube search pages are dynamic in nature and we can capture only upto 20 links using python requests package. To overcome this issue, we used python selenium libraries' web driver package. The selenium is used to automatically test webpages and in our project, we used it to scrape the links automatically by making a python script. After getting the links we used Beautiful Soup to scrape those links for titles and descriptions of those videos.

A. Selenium

Selenium is a framework widely used for software testing. Its' python library has the added benefit of being able to control web browsers like Chrome, Firefox, etc. Seleniums' *webdriverpackage* supports web pages that are dynamic in nature, i.e. the pages that can change their content without reloading[4]. Scraping dynamic web pages cannot be done by making a simple http request from python requests package wherein comes selenium. Once web page with search results are loaded in the browser window, a simple python script can be used to scroll to the end of the web page, effectively loading all of the search results for our query. Then video links can be scraped easily by using *get_elements_by_xpath()* method of the *webdriverclass*.

B. Python Requests

Python Requests Library is used to make HTTP requests in python. It can also be used to customize HTTP requests according to the needs of the user. To deal with HTTP requests in python is not a straight-forward task. It requires a lot of effort and knowledge to understand them. The basic library in Python to handle requests is *urllib*. *Urllib* is no doubt an efficient package in Python to handle HTTP requests. It can handle HTTP post and get requests. But to understand various methods and calls in *urllib* can be quite overwhelming to a fresher in programming. Python requests library is quite easy to use. It is user-friendly and uncomplicated than other libraries. By just simply calling the *requests.get()* method and passing in a url in the method we get a 'get' response of the webpage. It can be used to extract any kind of information from the web page that you want to. A drawback of requests package is that it does not perform well on dynamic web pages as they require human input[5]. The requests package was used in conjunction with BeautifulSoup to scrape the titles and descriptions of the videos we scraped with selenium.

C. BeautifulSoup

Beautiful Soup is a python library which can be used to scrape data from a website. It has the ability to fetch data from an HTML or XML format. Nowadays all websites are built under this format. We can apply different functions to filter out the web page for different tags, classes, IDs', attributes, etc. This is made possible due to the fact that BeautifulSoup generates a parse tree based on Document Object Model (DOM). Moving on to the working, the process involves passing the HTTP GET response to the *BeautifulSoup()* function which then converts elements of document into UNICODE characters. These characters are parsable and we can extract information from them as easily as from any other data structure in python. Therefore, Beautiful Soup library saves up a lot of time and reduces the effort of programmers.

Figure 2 explains our scraping engine. The webpage with html, css and JavaScript is fetched using seleniums' *webdriver* and requests package. Then the information is extracted using BeautifulSoup and stored in a csv (comma separated values) file.

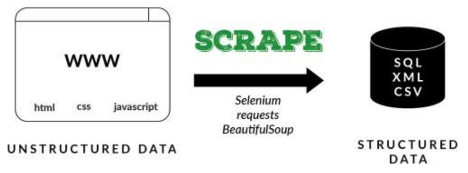

Figure 2. Scraping Engine

III. DATA CLEANING

The data cleaning process includes some basic paradigms used in cleaning process. First all of the text was converted to lowercase in order to make it consistent, then eliminating stop words and replacing words with their stem forms using the *Porter Stemmer*. We cannot forward plain text into the classifier[6]. In order to make it possible to classify the text, bag of words approach was used to feed data into the classifier.

A. Stopwords Removal

In natural language processing, stop words are words which are removed before processing of a body of text. There is no particular list of stop words, different industries can have different stop words. But the most common stop words include is, as, the, which, are, etc. For some cases stop words can be the most frequently occurring words, for others it can be some particular words with no meaning given a context. Stop words removal is done to reduce the dimensional space required when using *Count Vectorizer*method to make the Bag of Words. Sometimes omitting stop words can also lead to loss of context when semantic analysis of the text body is needed. In our case, the context doesn't really matter as we require to classify a body of text according to the words present in it and removing stop words isn't going to affect the performance of our classifier. We are using English stop words given in the *nltk* python library to remove the stop words.

B. Porter's Stemmer

To satisfy a user's request for information, the main concern is to select which piece of text in a collection should be fetched. The fetching decision is made by comparing the words of the sentence with the index terms (important words) of the document[7]. Basically, the motive by Word stemming is to reduce words to their stem form which enables to broaden the results by including both word roots and word derivations. Many words can have the same semantic interpretations and therefore a number of stemming algorithms were developed which help to reduce a word to its root form[7]. One of these algorithms is Porter's Stemmer Algorithm.

The Porter Stemmer was developed by Martin Porter in 1980 at the University of Cambridge. It is widely used in Natural Language Processing. The algorithm consists of 5 steps as shown in figure 3.

1) The *first step changes plurals and past participles into simplified form. For example:*

- Watched → watch
- Reading → read

2) The second step deals with pattern matching on common suffixes. For example:

- Happiness → happi
- Fractional → fraction
- Callousness → callous

3) The step 3 in this process deals with special word endings. For example:

- Duplicate → dupli
- Hateful → hate

4) Step 4 checks the processed word against more suffixes in case the word is a compound word. For example:

- Revival → reviv
- Allowance → allow
- Inference → infer

5) The last step, step 5 checks if the word ends with a vowel and fixes it. For example:

- Probate → probat
- Sulphate → sulphat

The algorithm is careful enough not to remove suffix from small words with smaller length stems.

Figure 3 shows the process of stemming the words to their *stem* form. It involves all the 5 steps explained above.

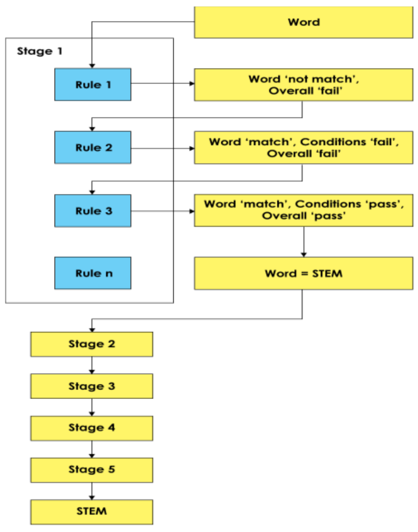
Word
Stage 1
Rule 1
Word 'not match', Overall 'fail'
Rule 2
Word 'match', Conditions 'fail', Overall 'fail'
Rule 3
Word 'match', Conditions 'pass', Overall 'pass'
Rule n
Word = STEM
Stage 2
Stage 3
Stage 4
Stage 5
STEM

Figure 3. Stemming Process

C. *Bag of Words*

Bag of Words technique is used widely in text classification and it has proven to be increasingly effective too. It is a technique where the words in the given corpus are counted and the number of occurrences of words are used instead of the whole sentence[8]. Two objectives are achieved by this approach:

- The size of all the textual data is consistent. It can now be fed into the classifier.
- The information from the sentence is extracted successfully.

We made a bag-of-words of 1500 words using the *CountVectorizerclass* from *sklearn.feature_extraction.text* package.

IV. RANDOM FOREST

Random Forest is an ensemble learning model proposed by the American scientist Leo Breiman for classification, regression and other methods which operates by constructing decision trees at the time of training. Decision trees are a popular method for various machine learning tasks. Random forest, as its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest gives out a class prediction and the class with the most votes becomes our model's prediction. It is based on the K decision tree $\{h(X,\theta_k), k = 1, 2, \ldots, K\}$ as a basic categorizer. The categorization result of the Random Forest is decided by each decision tree through a simple voting way. Furthermore, trees which are grown very deep tend to learn highly irregular patterns which means they overfit their training sets.

Figure 4 explains the process of prediction by random forests. We can see the 9 different decision trees giving different responses[9]. The prediction is made to be 1. The final output is calculated considering the five 1's and four 0's as a combined answer by all the decision trees. This is the basic concept of ensemble learning.

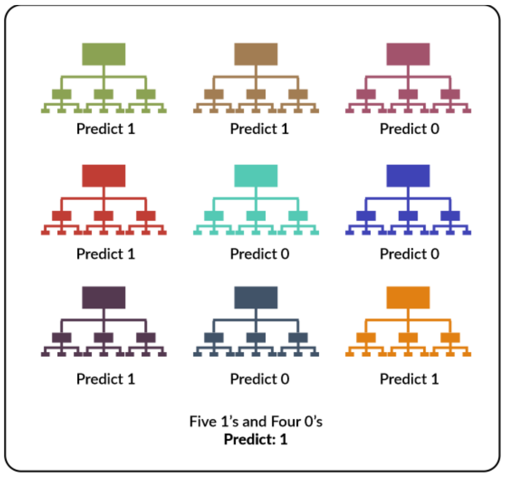

Figure 4. Prediction by Random Forest

While constructing the decision trees, the training set and attribute subsets are selected randomly. Decision trees are very sensitive to the data on which they are trained as any small changes to the training data can result in dramatically different tree structures. Random forest takes advantage of this by allowing each individual tree to randomly sample from the dataset with replacement, resulting in different trees. The random selection of data can be done in two ways:

*1) **Feature Subspace Idea:** During the construction of Decision Trees, an attribute subset needs to be extracted from all the nodes' attributes. To split the node, an optimal attribute is used.*

*2) **Bagging Approach:** Bagging technique is defined as the process of selecting random samples for all the different decision trees.*

In the process of construction of a Random Forest, all the samples and attributes for all Decision Trees are selected at random[10]. Hence, it gives RF a very strong generalization ability.

Figure 5 shows how a decision tree is constructed considering all the steps involved in a loop.

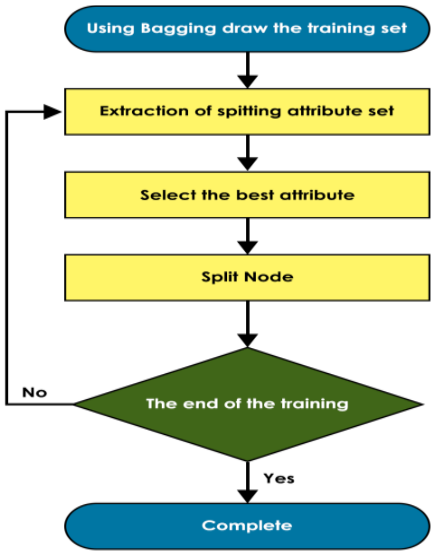
Using Bagging draw the training set
Extraction of spitting attribute set
Select the best attribute
Split Node
The end of the training
No
Yes
Complete

Figure 5. Construction of a Decision Tree

V. TOOLS USED

A. Python 3.7

Python is programming language which is readily available and solves computer problems by providing a simpler way to solve the problem. It's also known as a scripting language. Python is platform independent which makes it very versatile for programmers. Python has many libraries especially for data science and machine learning models. Before Python came into existence, ABC Language was used among the developers but python took its position right after landing in the market. Besides everything, Python is very famous among the developers as it is a dynamic language and also supports garbage collection[11-14]. One most important thing to remember while using Python is that keep a check of indentation, as one extra whitespace can lead to different outputs.

B. Natural Language Toolkit

Natural Language Processing Toolkit (NLTK) is a library in python that works with human language for natural language processing. Natural language processing works on statistical learning and machine learning and by using general learning algorithms it can interpret human language data[15]. Natural Language processing toolkit provides the processing and classification of text which is a very important part in the process of building model. It has many libraries and functions which can be used for tokenization, stemming, stop words, tagging, semantic reasoning and parsing. This toolkit was mainly created to extend the research in Natural Language Processing[16]. NLTK includes number of corpora and lexical sources for example, Problem Report Corpus, Movie Review Corpus and Lin's Dependency Thesaurus.

C. *Scikit-learn library*

Scikit-learn library is the most useful library for building models by data mining and data analysis in Python. It is an open source library in python. Built-on matplotlib, NumPy and SciPy, Scikit-learn library contains many simple and efficient tools for machine learning and statistical modeling including clustering, regression and classification. Some of the features of this library are cross-validation, feature extraction (like bag-of-words), supervised and non- supervised learning algorithms[17]. It is an open-source library for python and has various classification, regression, clustering algorithms[18]. It also consists of SVM (Support Vector Machine), random forests, gradient boosting, k-means. Sci-kit Library has support for numpy and scipy libraries too. Scikit library is originally written in python and some of the code written in Cython so that performance and efficiency can be achieved.

VI. Experimental Analysis

Data scraped from YouTube is used to train the RF Classifier. The size of the complete dataset is around 3200, out of which 20% is used for testing the model.

The precision rate (Precision) and recall rate (Recall) are used to evaluate the performance of a classifier.

$$Precision = \frac{TP}{TP + FP} * 100\% \tag{1}$$

$$Recall = \frac{TP}{TP + FN} * 100\% \tag{2}$$

Where TP, FP, FN are True Positive, False Positive, False Negative respectively.

A better way to evaluate the accuracy of categorization models is F-1 Score. It is calculated as:

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3}$$

The average *F1-Score* of our model was calculated to be 0.98. This is a really good score for a classifier. An *F-1 Score* of 0.98 means that predictions done by the classifier are 98% correct. The detailed classification report consisting of all Precision and Recall values can be seen in Table 1.

	PRECISION	RECALL	F-1 SCORE
0	0.95	1.00	0.97
1	1.00	0.95	0.97
2	0.97	0.99	0.98
3	1.00	0.98	0.99
4	0.99	0.99	0.99
5	0.98	0.97	0.98
accuracy			0.98
macro avg	0.98	0.98	0.98
weighted avg	0.98	0.98	0.98

Table 1. Classification Report

The above table shows all the calculated values from the classifier. We can see individual *Precision* and *Recall* values for the classes as well as their *F-1 Scores*. The final score of the classifier is calculated by taking an average of *F-1 Scores* of all the classes

VII. Conclusion

This paper first introduced data collection process using various python libraries and the methods which were used to collect the data. *Selenium* was used so that automated web scraping could be done and the data was fetched automatically by a script written in python. Then that collected data was exported to a csv file. Following which the most important feature of the project which was data cleaning consisted of removing stop words, using *Porter Stemmer* to strip each word to its stem and then making the bag of words using *CountVectorizer*. Then we had to label encode the target variable because classification model can't understand words as labels, so we used *LabelEncoder()* to encode the category which converted the words into numbers. It came to our notice that Random Forest is a famous and effective ensemble learning algorithm with strong performance in categorization. Therefore, in building the model and classifying the data we used Random forests. Finally, after initializing our model we split the dataset into train and test dataset. The model was then trained using training dataset and then testing was done with the test dataset. After which various evaluation metrics were used to evaluate the performance of our model and it turned out to be 98% accurate.

References

[1] Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent Trends in Deep Learning Based Natural Language Processing [Review Article]. IEEE Computational Intelligence Magazine, 13(3), 55–75. doi: 10.1109/mci.2018.2840738

[2] Babu, V. (n.d.). Web Scraping and Data Analysis using Selenium Webdriver and Python. Retrieved from https://www.datasciencecentral.com/profiles/blogs/web-scraping-and-data-analysis-using-selenium-webdriver-and

[3] C. Zheng, G. He, and Z. Peng, "A Study of Web Information Extraction Technology Based on Beautiful Soup," vol. 10, no. 6, pp. 381–387, 2015.

[4] Uppal, N., & Chopra, V. (2012, May). Design and Implementation in Selenium IDE with Web Driver. Retrieved October 5, 2019, from https://pdfs.semanticscholar.org/1127/8cc3164514146253ede9ecb6771f045fea0e.pdf.

[5] Dong Shishi, Huang Zhexue, "A Brief Theoretical Overview of Random Forests [J]", *Integrated Technologies*, vol. 2, no. 1, pp. 1-7, 2013.

[6] Xue, D., & Li, F. (2015). Research of Text Categorization Model based on Random Forests. *2015 IEEE International Conference on Computational Intelligence & Communication Technology*. doi: 10.1109/cict.2015.101.

[7] Issac, B., & Jap, W. J. (2009). Implementing spam detection using Bayesian and Porter Stemmer keyword stripping approaches. *TENCON 2009 - 2009 IEEE Region 10 Conference*. doi: 10.1109/tencon.2009.5396056

[8] Fernández, O. C. (2015, December). Web Scraping: Applications and Tools. Retrieved August 5, 2019, from https://www.europeandataportal.eu/sites/default/files/2015_web_scraping_applications_and_tools.pdf

[9] Weinstock, C. B., & Goodenough, J. B. (2006). On System Scalability. doi: 10.21236/ada457003

[10] Sirisuriya, D. S. (2015, November). A Comparative Study on Web Scraping. Retrieved August 4, 2019, from http://ir.kdu.ac.lk/handle/345/1051.

[11] Dean, J., & Ghemawat, S. (2008). MapReduce. *Communications of the ACM*, *51*(1), 107. doi: 10.1145/1327452.1327492

[12] 5 Tasty Python Web Scraping Libraries. (2018, February 8). Retrieved from https://elitedatascience.com/python-web-scraping-libraries.

[13] Garfinkel, S. L. (n.d.). An Evaluation of Amazon's GridComputing Services: EC2, S3, and SQS. Retrieved August 5, 2019, from http://nrs.harvard.edu/urn-3:HUL.InstRepos:24829568.

[14] Optimus Information. (2017, February 20). Selenium Testing: Advantages and Disadvantages. Retrieved from https://www.optimusinfo.com/blog/selenium-testing-advantages-and-disadvantages.

[15] Bahrami, M., Singhal, M., & Zhuang, Z. (2015). A cloud-based web crawler architecture. *2015 18th International Conference on Intelligence in Next Generation Networks*. doi: 10.1109/icin.2015.7073834

[16] Yadav, M., & Goyal, N. (2015, September). Comparison of Open Source Crawlers- A Review. Retrieved from https://www.ijser.org/researchpaper/Comparison-of-Open-Source-Crawlers--A-Review.pdf.

[17] Gandhi, V. A., & Kumbharana, C. K. (n.d.). Comparative study of Amazon EC2 and Microsoft Azure cloud architecture. Retrieved August 16, 2019, from http://www.ijana.in/Special Issue/22.pdf.

[18] Staff, F. T. (2018, November 30). 4 Infrastructure Requirements for Any Big Data Initiative. Retrieved from https://fedtechmagazine.com/article/2016/12/4-infrastructure-requirements-any-big-data-initiative.