Report for YouTube Model

BY: ABHINAV MITTRA

HOW I PREPROCESSED DATA?

I have used both beautifulSoup a python library for scraping webpages and Youtube Data API v3 to scrape the id, title, description and category name from youtube and then save it in a csv file. After scraping information for all the 6 categories I concatenated all the csv files into one with a total of 10200 records which I have used for my models. Both the codes have been written by me from scratch. For preprocessing of the data I have used regular expressions to remove urls, non-ascii text, punctuations, redundant whitespaces, @ symbols and the top 50 rare words to reduce dimensionality for text classification. For Text Normalization, I have used a PorterStemmer from nltk and a lemmatizer from textblob.

To represent text data as a valid form of input to my model's I used a TfIdfVectorizer(). I then applied K-means clustering to quickly visualize my data and choose a suitable algorithm.

WHY I CHOSE MY MODELS?

I chose SVM because it can model non-linear decision boundaries. And has kernels to choose from. Linear mostly fits well for text-based problems. They are also fairly robust against overfitting and is computationally cheaper.

I chose XGBOOST because it is highly flexible and versatile and can work through most classification and regression problems as well as user-built objective functions. It has proven to push the limits of computing power for boosted trees algorithms as it was built and developed for the sole purpose of model performance and computational speed. It is also robust to outliers in the inputs and immune to dimensionality.

I chose a LSTM model because they learn the order dependence between items in a sequence very well and are explicitly designed to avoid the long-term dependency problem. They also work extremely well for non-linearities in the data.

WHY I GOT THESE RESULTS?

The dataset that was formed by me was basically the first 1700 results scraped for each category and because youtube also indexes its videos according to keywords so most of them had the category name somewhere in their title or their description so it turned out to be a very easy text classification problem and hence the accuracies touch the pinnacle.

However to create a better dataset we can actually start from the last page of youtube and start scraping in reverse to make it a much more challenging problem. Since I already have my semester final exams going on I couldn't find the time to do this but I find the idea interesting and I'll definitely try it once my exams get over.

Results:

Support Vector Classification:

Test Accuracy: 99.950%

Classes	Precision	Recall	F1-Score	Support
Art&Music	1	1	1	669
Food	1	1	1	698
History	1	1	1	674
Manufacturing	1	1	1	673
Science&Technology	1	1	1	700
Travel	1	1	1	666

XGBOOST:

Test Accuracy: 99.975%

Classes	Precision	Recall	F1-Score	Support
Art&Music	1	1	1	684
Food	1	1	1	654
History	1	1	1	704
Manufacturing	1	1	1	659
Science&Technology	1	1	1	686
Travel	1	1	1	693

LSTM:

Test Accuracy: 99.438%

Classes	Precision	Recall	F1-Score	Support
Art&Music	1	1	1	314
Food	1	1	1	308
History	1	1	1	348
Manufacturing	1	0.99	0.99	358
Science&Technology	0.97	1	0.98	316
Travel	1	0.98	0.99	315