Predicting Underlying Themes and Projecting Popularity of Speeches

Fahmid Morshed Fahid, Nitesh S. R. Kovvuri, Rishal Shah
Department of Computer Science
North Carolina State University
Raleigh, NC 27695
`{ffahid, nkovvur, rshah27}@ncsu.edu`

1 Introduction

Speeches play a pivotal role in inspiring and galvanising individuals and have the power to reach the masses. The topic, content, and delivery style directly contribute to the popularity or the number of views for the speech. That is why it is valuable to predict these factors to understand how speeches resonate with the audience.

From human judgement, the subject matter of the talks or the speaker themselves are often seen to be the driving factor behind their acclaim. This project focuses on implementing a textual analyzer that combines other numerical facts of the data to learn the fundamental themes included in a formal address. The solution we propose utilizes concepts across Natural Language Processing and other data mining techniques to analyze transcripts of TED talks. Many popular approaches are explored across the data mining pipeline to answer the following research questions:

- **RQ1:** Can we predict view counts of a talk (Regression)?
- **RQ2:** Can we predict the type of a talk(Classification)?
- **RQ3:** Can we predict the popularity of a talk (Classification)?

The rest of the paper is organized as follows: Section 2 goes over various related work and background knowledge related to this work, section 3 describes the dataset in detail, section 4 describes all the methods which are used in the experiment and section 5 describes the experiment setup itself. Sections 6 and 7, discusses the results of the experiment and concludes with appropriate findings.

2 Background

Technology, Entertainment and Design (more popularly known as TED) talks are international conferences which provide platforms to individuals who have excelled at their respective fields of work to share their experiences and discoveries to a global audience.TED talks have become a household name since their inception and are also one of the most popular pieces of content published on the Internet, consistently in the order of millions of views. Hence, the popularity of TED talks may be a great way to gauge the interest of the general public and serves as an ideal dataset for the prediction task. There have been recent efforts which specifically make use of TED talks as the data-set to predict the impact of speakers on the viewer [Sugimoto et al. (2013)]. The authors found correlations and lack thereof between the presenter and the talk's popularity as well as citations received by the presenter in the scientific community after giving the talk. The work described in this paper instead does not consider the presenter when predicting the popularity but makes use of the content of the speech itself as a possible driving factor for its popularity.

Lewis et al. in their comparative study showed that Naive Bayesian Classifier and a constrained Decision Tree both are equally suitable for text classification [Lewis and Ringuette (1994)]. Using a global evaluation measure, the authors discard irrelevant features and demonstrate the contextual feature selection mechanism utilized by decision trees could be speed up through a powerful feature selection technique. On the other hand, empirical as well as theoretical evidence show that, Support Vector Machines (SVMs) are significantly better at text classification when compared to other traditional classifiers [Joachims (1998)].

In paper [Szabo and Huberman (2008)], the authors have proposed a method to predict the popularity in terms of views of user generated content soon after the submission of the video to streaming websites. They have implemented this by observing the popularity at an early time and then performing applied 3 models name - linear regression on a logarithmic scale, Constant Scaling model and Growth profile model. They found strong correlation between the logarithmically transformed popularities at early and later times. The accuracy of prediction showed a large dispersion around the average, when taking a direct squared error measure as compared to choosing relative errors. Thus leading to the conclusion that relative measures must be favoured over the absolute error measures.

The authors of [Nair et al. (2018)] demonstrate an algorithm which achieves a faster way of finding the optimal configuration of a software through by modeling the process using a sequential-model based method in lieu of a Gaussian Process Model. The algorithm takes into consideration the prior configuration state of the software system and makes a decision by sampling from the unexplored configuration space. To improve the scalability of the solution, the authors have made use of a decision tree learner. Various hyperparameters such as the minimum number of samples for leaf nodes, the depth of the tree, minimum number of samples for deciding a split etc. can be tuned using this method. For any classification task, this would reduce the computation time during hyperparameter optimization.

3 Methodology

As there are no state-of the art work, we experimented with different approaches in the data-mining pipeline and this section describes our approach in detail and the various methods which we have implemented.

3.1 Preprocessing

We first normalized all the numerical features, so that the variations within each feature are comparable across all the columns. We made use of Min-Max normalization for our dataset:

$$normalizedValue = \frac{x - max}{min - max} \tag{1}$$

Next, we converted all text features into weight vectors and combined them with existing numerical features. For deriving the word vectors, we have used TF-IDF using the following equation:

$$idf_{d,t} = \log[\frac{1+n}{1+df_{d,t}}] + 1 \tag{2}$$

$$tf - idf_{t,d} = tf_{t,d} \times idft, d \tag{3}$$

where, n is the total number of documents, $df_{d,t}$ is the document frequency for term t in document d and $tf_{t,d}$ is (the number of times term t appears in document d) / (total number of terms in document d).

Next, we performed feature selection by using PCA on all the attributes. We plotted different graphs to see the variation for different principle components to select the optimum number of components which we feel captured the most amount of variance.

3.2 Prediction Models

We have used both regression models and a classification models, to train over our dataset. We target views for regression tasks and popularity and talkType for the classification tasks.

3.2.1 Classification Models

To correctly classify new data points using the target features popularity and talkType, we make use of three types of classification models. Namely:

Decision Tree Classifier Decision trees based classification builds a conditional tree dynamically when it trains over the data. At each decision point on the tree, it tries to capture the most information which can be derived from the dataset. Information captured or the purity of the data can be measured using various metrics such as Information Gain and the Gini Index. The Gini Index was chosen as the criteria for the decision for this experiment.

Support Vector Classifier Support Vectors based classifiers are very robust and can tolerate outliers as well to a certain extent. The decision boundary between the data points is allocated in such a way that the minimum distance from a datapoint of either class to the separation hyperplane is maximized. To tolerate outliers, Support Vector classifiers also take additional parameters like the penalty factor.

Random Forests This is an ensemble method which generates multiple decision trees, which could be weak learners individually and weights the answer of each tree to create a strong learner. It is also important to note that for every decision tree in the ensemble a random subset of the feature set is used to build the decision tree.

3.2.2 Regression Models

For the views prediction task, we have implemented a variety of regression models to project the data and compare all their performances. The following are the various models we used:

Decision Tree Regressor A decision tree regressor works similar to the decision tree classifier. But in this case, the prediction is the weighted average of the data available at the node and the purity measure of the continuous data at the node can be measured using the mean squared error value.

Support Vector Regression Support Vectors based regression uses similar mathematical instruments as Support Vector classifiers. But the optimization problem it tries to solve is that the distance between all the data points to the hyperplane is minimized and that the hyperplane is as flat as possible. SVRs also have penalty factors to take care of outliers in the data.

Linear Regression A linear function is closely fitted to the training data points by using the Ordinary Least Squares method, which minimizes the sum of the squared distance of all the data points from the fitting line.

Ridge Regression Ridge regression is essentially a constrained linear regression which limits the range of the coefficients for the independent variables so as to avoid overfitting to the training data. Ridge regression models generally have a hyperparameter like the degree of fit to control the fit to the training data set.

3.3 FLASH Hyperparameter Optimizer

FLASH is a Sequential-Model Based Optimizer (SMBO) where it uses an acquisition function to predict the next best hyperparameters. It builds a decision tree regressor using the current pareto frontier of configurations and based on the target function (in our case, F-Score), it retrieves the best configuration. This can be used to tune any classifier with multiple hyperparameters.

3.4 Performance Measure

For measuring the performance on our test data (using k-fold) on the regression task, we have used Mean Square Error (MSE) against the ground truth for number of view counts (Views). Our target was to minimize the MSE. The following equation was used for calculation.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{4}$$

Our classification task classifies our dataset in 10 categories and all the categories are equally important. Since this is a multi-class classification, we calculate a macro average of Precision, Recall for calculating a macro average of F1 Score. The following equations were used to evaluate individual performances:

$$Precision = \frac{tp}{tp + fp} \tag{5}$$

$$Recall = \frac{tp}{tp + fn} \tag{6}$$

Macro averages are represented mathematically as:

$$Macro_X = \frac{\sum_{c \in Category} X_c}{\sum_{c \in Category} 1} \tag{7}$$

And for calculating F1 score, we have used the following.

$$F1Score = \frac{2 \times Macro_{precision} \times Macro_{recall}}{Macro_{precision} + Macro_{recall}} \tag{8}$$

We have set the objective for our models to achieve an optimum F1Score.

3.5 K Fold Croos Validation

K-Fold cross validation is resampling approach that is used to validate result on a limited dataset. For validation of our result, we have used a K-Fold cross validation approach. In this approach, the dataset is splitted into k groups, taking one group on each iteration as hold-out set and the rest as training set. The process completes this for each of the split and then the combined result is reported. We have used stratified K Fold to ensure the ratio is kept in our imbalanced dataset when doing classification.

For tuning hyperparameter, we further randomly splitted our training set into 9:1 as training and validation set. Note that, we did not include our test set when tuning hyperparameters.

4 Dataset

Initially, our dataset contained two separate files, one file (main.csv, 2550 rows, 17 columns) contains the details of each TED talk such as the speaker name, title, description, duration, number of comments, ratings, publication date, filming date, event name etc while the other file (transcript.csv, 2467 rows, 2 columns) contains the transcript of the talk itself. There are some missing values in the transcript file (that cannot be mapped with the main.csv file) and we have ignored them as noises.

After removing all unnecessary columns based on our intuition (such as speaker name, url etc) and combining the two files by throwing away incomplete data, we obtain our initial dataset. The processed dataset contains the following ten features: title, description, transcript, tags, comments, duration, event, language, num_speaker and published_date.

Desciption	Value
Numerical Feature Columns	5
Text Feature Columns	5
Datapoint	2467
Datapoint deleted (garbage)	83
Regression Target Column	Views
Classification Target Column	talkType
Classification Target Column	popularity

Table 1: Dataset description

Regression Task: Views	Value
Min	155895
Max	47227110
Mean	1740295
Median	1149090
std	2527086

Table 2: Regression detail for column Views

For our regression task, we have used the Views column as our target column (see table 2), while for the classification task, we have modified the ratings column and created a new feature called talkType by finding the highest valued rating for each row. We combined all the ratings type that has less than a count of 40 in the Other category (see table 3). This was an engineering choice. For another classification task, we created a new column called popularity that was calculated using the quartile values of views per year. Talks having less than 25% view in a corresponding year are unpopular, while between 25% to 75% are moderately popular and others that were above 75% are extremely popular (see table 4). Again, the choice of 25% and 75% were engineering judgement.

A short description of our dataset is given in the table 1.

Category	Count
Inspiring	851
Informative	711
Fascinating	251
Funny	159
Beautiful	142
Ingenious	101
Courageous	82
Persuasive	79
Jaw-dropping	49
Other	42

Table 3: Support for each category in TalkType

Category	Count
Very Popular	618
Moderately Popular	1231
Unpopular	618

Table 4: Support for each category in Popularity

Note that, we have not split our data into test and training because we are using K-fold cross validation.

5 Experiment Setup

We designed our experiment using a 10-fold cross validation approach. Note that, for classification task, our target column is skewed, thus we have used a stratified approach. For regression task, we have used a regular K-fold classifier. Also, for optimization, we further splitted the training set into 9:1 for training and validation. All our experiment has been run 10 times and only the median values are being reported. We ignored mean as they are prone to outliers. Next, we normalized the numerical columns using our training set only. This is done to give each feature equal weight. For our text columns, we have used a TF-IDF vectorizer. To this end, we have used different max number of words (engineering decisions) for each text attribute (see table 5 for details). For text vectorization, we have converted all text to lowercase, removed punctuation and english stop words and used l2 normalization. We have merged all the text-vectors in our feature list (one feature for each word in each text) and dropped the original texts.

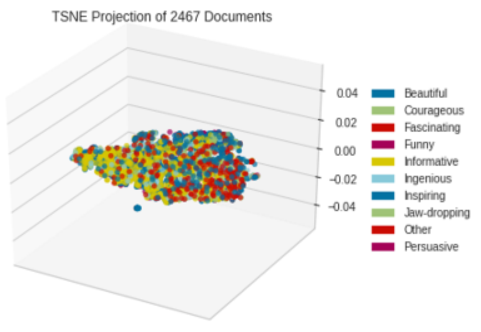
Text Feature	Max Words
transcript	2000
description	200
title	100
tags	100
event	10

Table 5: Text Feature Vectorization

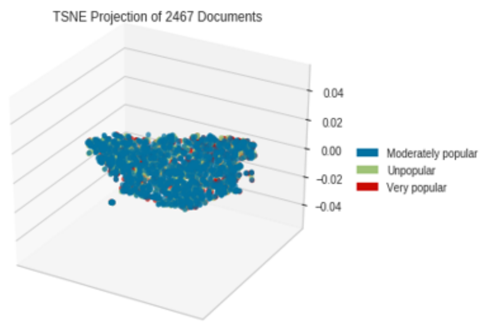
Item	Specific parameters
K-fold	k=10, shuffled, stratified
TF-IDF Vectorizer	lowercase, english stopword, l2 norm
PCA	cov-matrix eigenvalues, 600 components
Linear Regression	Include intercept
Decision Tree	gini index, best split, 2 samples for split
SVC (Untuned)	linear kernel, C=100, auto gamma
Ridge Regression	Degree of constraint α = 2
SVR	C=1, auto gamma, rbf kernel

Table 6: Different choices for experiment

We plotted the processed features using the t-distributed Stochastic Neighbour Embedding algorithm to visualize vectors in lower dimensions by preserving the relative distances between all the vectors when projected. From Figure 1 it is immediately apparent that the word vectors are not linearly separable in lower dimensions. Hence, separating the features to successfully classify according to the target column, would be a hard task. This would mean that there is no one word which would certainly contribute to its popularity or the talk type.



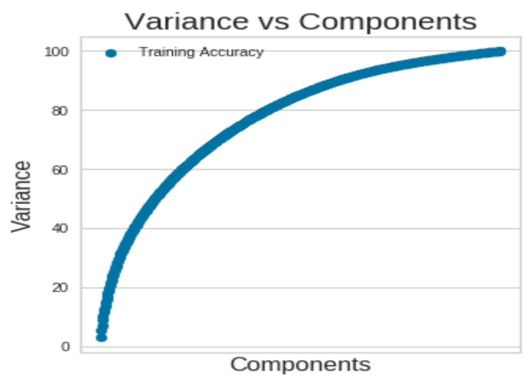
(a) Visualization using talkType label



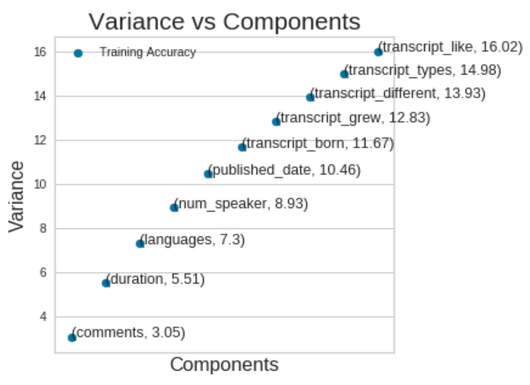
(b) Visualization using popularity label

Figure 1: Word Embedding Visualization in 3 dimensions

We then make use of the top 2410 (2000+200+100+100+10, see table 5) most important word vectors and 5 numeric features upon which we apply PCA to visualize the captured variance in features of the training set and found that top 600 features accounts for more than 90% variance in the data. We thus selected the top 600 features for our prediction. We can see on Figure 2b that the numerical features seem to capture more variance than word vectors, attributable to word vector matrices being sparse.



(a) Variance versus all the components in order of variance captured



(b) Variance versus first 10 components in order of variance captured

Figure 2: Component Variance Analysis

For regression, we trained a Linear Regression Model [Szabo and Huberman (2008)], a support vector regressor and ridge regressor. For classification, we have trained a Decision Tree Classifier, Random Forest Classifier and a Support Vector Classifier [Joachims (1998); Lewis and Ringuette (1994)]. The choice of classifiers were based on the empirical results for text classification. A detailed choice with parameters are given in table 6). We then evaluated our model in our test set using K-fold cross validation. After doing some initial experiments, we have found that, SVM does significantly better in both of the classifiation tasks. Thus we further tuned the parameters of SVM using a hyperparameter optimizer called FLASH with a initial budget of 10 and initial pool size of 10. We found that, for most training set, rbf kernel with c value around 10 performs better. We kept this embedded in our experiment for new training and validation set. We call it SVClassifierTuned.

6 Results

This section describes our model performance comparisons and other findings. The results presented here answer the questions posed before in section 1.

RQ1: The view count for the speech could be extrapolated to some extent. The best performing regression model over the test data was ridge regression. This may imply that overfitting to the training data cost the model accuracy. It should also be noted that the Root Mean Squared Error calculated for this model was 2247950.23.

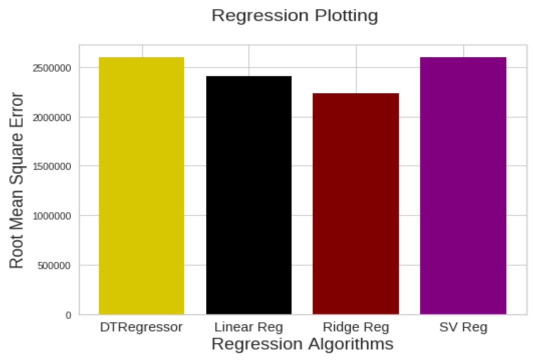
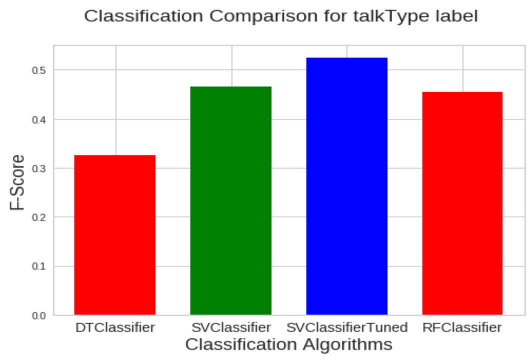


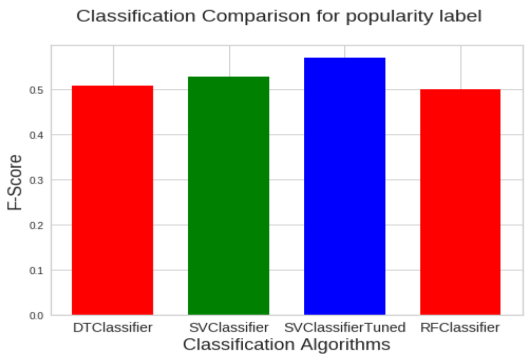
Figure 3: Comparison of regression algorithms

RQ2: We observed that the Support Vector Classifier with an arbitrary penalty factor of 100 and a linear kernel seems to outperform the other two classifiers we considered for the talkType label with an F1-score of 0.5269. Tuning the hyperparameters of the Support Vector Classifier using FLASH seems to have boosted its performance even more giving an F1-score of 0.569, an increase of 6.8%.

RQ3: For the classification task of talkType target, we observe the Support Vector Classifier again outperforming the other models with an F1-score of 0.465. The performance is boosted with hyperparameter tuning, which yields an F1-score of 0.5245, an improvment of 12.7%.



(a) Classification using talkType target



(b) Classification using popularity target

Figure 4: Performance comparison of classifier models

We also observe that Principal Component Analysis makes a difference in the classification task. If we do not perform PCA and retain all 2400 components to train the model with, we see a difference in F1-score for both *popularity* and *talkType* targets with scores of 0.564 and 0.573 respectively. We do a see slight decrease in the score for the *popularity* column, attributable to overfitting with many features.

7 Discussion

Our result show that, we might be able to predict the talk type and the polularity of a speech given its metadata and transcript, but it is significantly hard to predict the view counts in future.

Processing the view count column, we see that it has a standard deviation of 2527086, implying that it packs a lot of variance and may not necessarily be fitted using any model. This would explain the poor performance of our regression models, having very high RMSE. Among the models we tried, Redge Regression has the lowest RMSE of 2247950.23.

It is also important to note that the talkType column in the dataset was skewed. For example, there were 851 objects which were labelled as *Inspiring* as opposed to 42 labels for the *Other* category. This may have affected the performance of our classification models.

But, the models for classification perform fairly well for both target columns considered. We observe F-scores of over 0.5 for both talkType and popularity columns. This is much better than a random guess when we consider there are 10 classes and 3 classes for the target columns.

A significant increase in performance is seen when the classifier is tuned with FLASH. We observed a 6% to 12% increase in F1-Score. This indicates possible future improvements.

After deriving the word vectors, the number of total features of the dataset increased to over 2400. Principal Component Analysis was able to reduce it to a more manageable 600, but wasn't to able to reduce the dimensionality to a very large extent without compromising on variance.

8 Conclusion

In this work, we have performed data cleaning on our dataset and explored different techniques to preprocess the textual attributes. We have also performed feature selection using a dimensionality reduction technique like PCA over the weighted sparse matrix derived from the text data. We have implemented different data mining algorithms to do both prediction and regression. We also implemented hyperparameter tuning in our classification task for improvement. We have evaluated our approaches using different metrics.

Our result show promising outcome of predicting impact of speeches when combined with metadata of the event. Although, predicting the number of views is a hard task as there are many parameters which sometimes have no relation to the talk itself, we tried gathering and capturing as much information we could and implemented a number of classification and regression algorithms to fit our data into the model. Our work is not comprehensive and there are many other approaches that can be used. We deliberately omitted complex models (such as neural network) for this study and we encourage the reader to implement more sophisticated model and observe the outcome. For future scope we believe that if we can tune and optimize the hyperparameters for regression, then we could improve the RMSE values and thus improve its accuracy. Also we could improve the metrics of our models if we could implement and test Linear Discriminant Analysis for feature reduction in our pre-processing step. Finally, applying the concepts taught in class related to Artificial Neural Networks might provide us improved results over the current setup.

Note that, all of our work is available in at https://github.ncsu.edu/rshah27/SpeechImpactPrediction for reproduction purpose.

References

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*. Springer, 137–142.

David D Lewis and Marc Ringuette. 1994. A comparison of two learning algorithms for text categorization. In *Third annual symposium on document analysis and information retrieval*, Vol. 33. 81–93.

Vivek Nair, Zhe Yu, Tim Menzies, Norbert Siegmund, and Sven Apel. 2018. Finding faster configurations using FLASH. *IEEE Transactions on Software Engineering* (2018).

Cassidy R. Sugimoto, Mike Thelwall, Vincent Larivière, Andrew Tsou, Philippe Mongeon, and Benoit Macaluso. 2013. Scientists Popularizing Science: Characteristics and Impact of TED Talk Presenters. *PLOS ONE* (2013). https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0062403

Gabor Szabo and Bernardo A Huberman. 2008. Predicting the popularity of online content. *Available at SSRN 1295610* (2008).