Proceedings of The National Conference
On Undergraduate Research (NCUR) 2018
University of Central Oklahoma
Edmond, Oklahoma
April 5-7, 2018

Brigham Young University Speeches Popularity Predictor

Andrew Henrichsen, Ben Jafek, Jacob O'Bryant, McKell Stauffer
Computer Science
Brigham Young University
Provo, UT 84602

Faculty Advisor: Dr. Tyler Jarvis

Abstract

Natural language processing is currently an active area of research in machine learning. A particular subgroup of natural language processing strives to predict the "popularity" of a given text. Brigham Young University (BYU) maintains an archive of speeches that are given weekly on campus. We extracted a variety of features from the speeches and ran several machine learning models to predict which talks were popular (according to page views). All the models performed better than the baseline, but the J48 decision tree was the most effective. We did feature reduction using a wrapper algorithm, but this didn't improve the overall performance of our models. We discovered which features were most helpful by examining the decision tree's output.

Keywords: Natural Language Processing, Decision Tree, Machine Learning

1. Introduction

Every Tuesday morning at Brigham Young University, a speaker is invited to deliver an address to the university body. Their speeches are often religious in nature, but other topics range from athletics to chemistry. Over 2,400 of these speeches delivered from 1946 to the present have been archived on the university-sponsored website speeches.byu.edu. But many of the older speeches have not yet been put on the website, as they have yet to be edited. The editing process (e.g. the verification of sources, grammar fixes, and inclusion of visual supplements) could take a substantial amount of time. It is therefore important to the team that edits these speeches that the public would want to read them. For this purpose (and to be able to choose which speeches should be featured on the BYU speeches website and social media accounts), we decided to build a "popularity" predictor (a machine learning model to predict the number of page views that a speech would receive).

We decided to implement several different predictors, each created with a different machine learning algorithm. Particularly, we decided to focus on a Bayesian network, a J48 decision tree, a multi-layered perceptron, and a naive Bayes classifier. In this paper, we will compare these models using several different evaluation metrics. Additionally, we will attempt to improve the models' performance using feature selection. Our goal is to find a model that can be used for popularity prediction.

The outline of the paper is as follows: Section 2 will briefly explore past research into text popularity prediction. Section 3 will expound upon our methods. Specifically, it will discuss our data collection, cleaning and feature engineering. Additionally, it will introduce our training and feature selection models and their associated evaluation metrics. Section 4 will present and analyze our results. We will conclude in Section 5.

2. Related Work

Natural language processing is a vast field of research, full of myriad applications. In particular, we will focus this section on predicting the popularity of a given text. Several very different approaches have been taken to solve this problem. Zhang et al. propose a novel model which they name an "information diffusion model" to predict the popularity of events in microblogging (e.g. Tweets and Facebook posts) in the future[1]. Their model uses a dynamical system and context clues to model the spread of a microblog event. This model is similar to ours in the fact that the main features come from the text itself, and from information about the blogger/speaker.

More recently, researchers have been using machine learning algorithms to predict popularity. Bandari, Asur and Huberman used regression and different classification algorithms to predict the popularity of certain news stories in social media[2]. Using both linear and support vector machine regression, they found R^2 values around 0.33. So the regression models were effective enough to explain a significant amount of the variability of the data, but not fully. For the classification methods, they decided to focus on bagging, decision trees, support vector machines and naive Bayes. They found all to have high accuracies, with the highest being bagging at 83.9% followed closely by decision trees. This study is very similar to our own in the fact that it uses machine learning algorithms to solve the problem.

These two methods (dynamical systems and machine learning models) have been the main ways to study text popularity prediction in the past. We hope to complement the research that has already been done, by finding additional useful features and effective machine learning models.

3. Methods

3.1 Data Collection

Our data had several facets to it and was collected in several different ways. First, we scraped all of the speeches from speeches.byu.edu which had been viewed by at least 1 person in the last 18 months. We then created a useful dataset by extracting features from each speech (to see a list of features, please refer to Table 1). Additionally, we were generously given the Google Analytics data for all of the web pages of speeches.byu.edu from the BYU speeches team. From this data, we were most interested in keeping the organic page views for each web page (page views that did not come from social media links). This is because page views from social media links could become popular because an influential person shared the link, not because of the intrinsic properties of the speech.

We were particularly interested in developing an understanding of the features that were likely to lead to a high page view count. This understanding would allow the developers and editors who maintain the website to make informed proactive decisions about which speeches to advertise. This university-sponsored archive is very popular among its target audience, so the speeches which have a high page view count are culturally known. The most successful advertised speeches are those which contain desirable content but are overlooked by the public. Thus, we wanted to sort out those speeches which had a high predicted pageview count, but low ground-truth page view count. This false positive would suggest that the speech could become popular if advertised correctly. Thus, our model need not be perfect, but must err heavily on the side of outputting too many false positives rather than false negatives.

We stored our data in a SQLite database. We created a Python script to automatically check the database and compute only the missing features. This allowed us to incrementally add features without having to recompute the entire feature set. Finally, we exported the data from our database to a CSV file.

3.2 Feature Engineering

We engineered 23 features from our data, which can be seen in Table 1. In this section, we will detail how we engineered the least-obvious of these features. We calculated gender by extracting the speaker's first name from the URL associated with the talk. We obtained a list of baby names from the US Social Security database[3], with the number of males and females that have been given that name since 1880 in the United States. Thus, for each speaker's name, we classified it by the gender ratio (number of males over number of females). When the name was not in the list, the value "unknown" was put in as the gender value. For a few of these features, we classified them by hand (e.g. we classified Elder Lynn G. Robbins as male, as his name was classified as female on the list). In addition, some of the speeches were delivered and archived as a couple (e.g. Jeffrey and Patricia Holland's *Some Things We Have Learned — Together*). Consequently, we labeled the gender of these speeches' authors as "combo".

We calculated the polarity, subjectivity, and Flesch reading ease of the data using the natural language processing Python package Textblob[4]. The Flesch reading ease is a measure of the readability of a text as a function of average word length and average sentence length; it is calculated as

$$206.835 - 1.015 * \frac{\text{Total Words}}{\text{Total Sentences}} - 84.6 * \frac{\text{Total syllables}}{\text{Total words}}.$$

The references for the scriptures were found looking for book names preceding scripture citations in the text (e.g. the word "Genesis" immediately before a number such as 36:9 would imply that the scripture is an Old Testament reference). We determined if a reference was a scripture by looking at the format of the source reference. Specifically, the RegEx command searched for patterns of the form "Name ##:##". For authority mentions, we looked for the following phrases: "Prophet", "Elder", "President", and "of the Quorum of the Twelve".

For our output class, we created a feature called "popular." If the number of page views for a talk was in the 80th percentile or above, we classified it as popular. We originally looked at page views divided by days elapsed, since we thought that older talks would be favored to have more page views simply due to being available for more time. But we found that this method heavily skewed popularity in favor of recent talks, so we changed our output label to correspond with just the number of page views. We realize that this could add an amount of bias into our label, which could be fixed in future work. A better label might be based on the number of page views received in the first *k* months after the speech was published on the website.

3.3 Data Cleaning

As we created our own features, the only unknown or missing data arose with the gender feature. We decided to leave "Unknown" as a possible category for the gender classification, with 82 speeches falling into this category. We also had to deal with duplicate speeches, as often multiple URLs arise for each webpage as the result of programmer inconsistency (e.g. /talks/jarvis-tyler-j thats-light-gets/ v. /talks/tylerj-jarvis thats-light-gets/). Consequently, we had to identify and merge these speeches, summing their page view counts. Our final dataset had 1,516 different speeches. As can be seen in Table 1, the range of many of the features varied. We consequently decided to normalize each of our features.

3.4 Models

For our models, we decided to use a J48 decision tree, multilayer perceptron, Bayesian network, and a naive Bayes classifier. For each of these, we used the implementation found in the data mining softwareWeka. For our J48 decision tree, we set the pruning confidence to 0.25 and the minimum instances per leaf to 2. For our multi-layered perceptron we chose a learning rate of 0.3, a momentum of 0.2 (used in backpropagation), no early stopping via a validation set, a seed of 0 into the random number generator, the number of nodes in each hidden layer to be

$$\frac{\text{Attribute Count} + \text{Classes Count}}{2};$$

we trained the models for 500 epochs. For the Bayesian network, we decided to set the maximum number of parents to be 1, and the descendant population size to 0.5. Additionally, we decided to use a Bayes score, K2 search method, and a simple estimator, while not using the ADTree data structure. Finally, we used no special parameters for our naive Bayes implementation.

3.5 Evaluation

As stated above, the predicted uses for this model will be (1) deciding which speeches to professionally edit, and (2) deciding which speeches to highlight on the Speeches site or its social media pages. Thus, we are much more interested in avoiding Type 1 errors — we want the editors to be confident that if the model suggests highlighting a speech, then

that speech will be successful. We are less concerned about popular speeches being misclassified as not popular as long as enough speeches are correctly classified as popular so that the speeches team can find enough worthwhile speeches to focus their resources on. In short, we value precision over recall. The standard F_1 score given by

$$2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

weights both precision and recall equally. Since we value precision over recall, we decided to use the more general $F_ß$ score defined by

$$(1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

which gives ß times as much weight to recall as precision. Specifically, we use the $F_{0.2}$ score (recall that 20% of our talks are labeled as *popular*.). If we predict every talk as popular, the $F_{0.2}$ score would be approximately 0.206. This is our baseline. (If we predict every talk as not popular, then the $F_{0.2}$ score would be $\frac{0}{0}$ which we define as 0).

Also note that classification accuracy is an unsuitable evaluation metric for this application. For example, if we predict every talk as not popular, classification accuracy would be 80%. However, a model with a lower classification accuracy could easily have a higher $F_{0.2}$ score.

Table 1. This table contains a list of all of the features, along with their descriptions. It also contains the average values for each feature, including when they are classified as popular (within the top 20% of page views) or unpopular.

Attribute	Description	Popular	Unpopular	Overall
Gender	Gender of the speaker For these calculations we used Male: 0, Female: 1 We also threw out all other classifications	0.173	0.2214	0.2113
Polarity	Float $\in$ [-1, 1] Negative: -1, Neutral: 0, Positivity: 1	0.1688	0.1817	0.1791
Subjectivity	Float $\in$ [0, 1] Objective: 0, Subjective: 1	0.5070	0.5028	0.5036
Word Count	Number of words in a speech	4,703.20	4,150.75	4,264.78
Word Quantity	Number of unique words in a speech	1,337.52	1,220.99	1,245.05
Flesch Reading Ease	Readability of a text Int $\in$ {0, ..., 100} Hard to read: 0, Easy to read: 100	5.801	6.240	6.150
OT Ref	Number of Old Testament scriptures used.	1.497	1.229	1.285
NT Ref	Number of New Testament scriptures used	4.047	3.118	3.310
BOM Ref	Number of Book of Mormon scriptures used	4.872	3.411	3.713
D&C Ref	Number of Doctrine and Covenants scriptures used	2.875	3.162	3.103
Pearl of Great Price Ref	Number of Pearl of Great Price scriptures used	0.608	0.624	0.621
All Scripture Count	Number of scriptures used	14.679	10.974	11.739
Story Names	Number of common names referred to Does not include scripture references or authority mentions	14.402	12.978	13.272
Talking Speed	Word Count Length of audio file (seconds)	2.335	1.940	2.021
Authority Mentions	Number of references to authoritative figures	8.936	8.303	8.434
We You Ratio	(Number of times "We" appears)^2 Number of times "You" appears	142.985	137.302	138.475
First Person Pronouns	Number of times "I" appears	75.21	66.92	68.63
Percent in Italics	Percent of words in italics	0.0979	0.1025	0.1017
Percent in Quotes	Percent of words in quotation marks	0.00420	0.00510	0.00491
Days Since Delivery	Number of days between date given and November 19, 2017	6,784.55	7,682.51	7,497.15
Name Search Results	Number of hits on Google for a speaker's name	3,117,180	2,638,072	2,749,352
Year Given	Year that the speech was given	1998.9	1996.4	1996.9
Month Given	Month that the speech was given	5.87	6.28	6.19
Popular	Whether it above the 80-th percentile for page views No: 0, Yes: 1	1	0	0.20

3.6 Feature Selection

After evaluating our models using the $F_{0.2}$ score, we did feature selection to see if reducing the number of features could improve performance. We used two feature selection algorithms.

First, we used a wrapper algorithm to find an optimal subset of features. The algorithm started with no features and progressively added the features that most improved the F_1 score.

[1] For each subset of the features, the F_1 score was calculated using a J48 decision tree on the entire training dataset.

Second, we ran the same wrapper algorithm but using 10-fold cross-validation. In this case, the wrapper algorithm generated 10 optimal feature subsets, each time using 90% of the data for training and the other 10% for validation. The output was the number of times each feature was included in one of the 10 optimal subsets. We combined this into a final feature subset by choosing only features that were included in at least 5 of the optimal subsets.

4. Results

Table 2. This table contains the results for our models run on all of the features.

Model	$F_{0.2}$ Measure	F_1 Measure	Precision	Recall	RMSE	Accuracy
Bayes Net	0.357	0.276	0.366	0.221	0.4016	76.78%
Decision Tree (J48)	0.363	0.292	0.371	0.241	0.4493	76.65%
Multi-Layered Perceptron	0.305	0.258	0.310	0.258	0.4322	74.60%
Naïve Bayes	0.323	0.347	0.321	0.376	0.4416	71.64%

Table 3. This table contains the results for models run on only the selected features from the wrapper algorithm.

Model	$F_{0.2}$ Measure	F_1 Measure	Precision	Recall	RMSE	Accuracy
Bayes Net	0.344	0.274	0.352	0.224	0.3984	76.25%
Decision Tree (J48)	0.349	0.192	0.375	0.129	0.4186	78.30%
Multi-Layered Perceptron	0.339	0.216	0.356	0.155	0.3978	77.51%
Naïve Bayes	0.303	0.184	0.320	0.129	0.4159	77.11%

We first tested our models with all of the features. Our results are summarized in Table 2. The table includes a variety of different measures of success, to more completely compare the models. Notice that different models excel according to different measures. For our purposes, we are most interested in the $F_{0.2}$ score, so the best-performing model on the full feature set was the J48 decision tree. Although its recall is more than 0.1 less than the recall of the naive Bayes model, its superior precision contributes more to the $F_{0.2}$ score. If we were more interested in recall, the naive Bayes would be considerably better. If we were solely interested in accuracy and RMSE, then the Bayes net would be the best model. All of the models attained an $F_{0.2}$ score greater than 0.206, the baseline.

The optimal feature subset that the wrapper algorithm generated when running on the full training set was polarity, subjectivity, OT ref, BOM ref, Pearl of Great Price ref, all scripture count, authority mentions, word quantity, first person pronouns, percent in quotes, days since delivery, and year given. Table 3 shows the results when running the models on this subset. The 10-fold cross validation wrapper generated the subset polarity, subjectivity, OT ref, D&C ref, Pearl of Great Price ref, all scripture count, authority mentions, we-to-you ratio, first person pronouns, percent in quotes, days since delivery, name search results, month given, and year given. The results for this subset are given in Table 4. It is of note that the features that were most significant in both algorithms were: polarity, subjectivity, OT ref, all scripture count, authority mentions, first person pronouns, percent in quotes, days since delivery, and year given.

Notice that the first feature subset gives better precision in every case, at the expense of all of the other measures. We conjecture that this is because this subset is taken from a single run of the wrapper algorithm and thus retains features which "work well together," i.e. they partition the information space well. The second feature subset is an aggregation of features from multiple subsets, so they aren't as cohesive. Although the 10-fold cross validation reaches a peak accuracy of 78.89% and RMSE of 0.225 for the Bayesian network model, its precision of only 0.046 penalizes its $F_{0.2}$ score too much for it to be considered a useful model for this application. In fact, the $F_{0.2}$ score for each model when running on the second subset was less than 0.206, our baseline measure.

If we compare Tables 2 and 3, we can see that feature selection increased the J48 decision tree's precision slightly; however, the $F_{0.2}$ score decreased because recall was significantly reduced. The only model that actually benefited from feature selection was the multi-layered perceptron, but this model still didn't perform as well as either the Bayesian network or the J48 decision tree.

The model with the highest $F_{0.2}$ score overall was the J48 decision tree when trained on all features, so this is the model that we will present to the BYU speeches team. It is helpful that the decision tree was the most accurate, because it is also the most intelligible. In the decision tree that was generated, the three most important features were talking speed, word count, and year given, in that order. For example, our model predicts that if the speaker spoke more than 0.08 words per second, and fewer than 1,601 words in the year 2003 or later, then the speech would not be popular.

Table 4. These are our results for the different models using the selected features from the 10-fold cross-validation.

Model	$F_{0.2}$ Measure	F_1 Measure	Precision	Recall	RMSE	Accuracy
Bayes Net	0.080	0.311	0.046	0.3991	0.255	78.89%
Decision Tree (J48)	0.201	0.347	0.142	0.4284	0.329	77.51%
Multi-Layered Perceptron	0.188	0.376	0.125	0.408	0.349	78/36%
Naïve Bayes	0.120	0.250	0.079	0.4204	0.231	76.85%

5. Conclusion

We found that the J48 decision tree was able to achieve the highest $F_{0.2}$ score, at 0.363. This is considerably higher than the baseline score of 0.206, which would occur in the case of classifying every speech as popular. This is the model that we will present to the BYU speeches team. However, we do not anticipate that a 37.1% precision rate will be high enough to put the tool immediately into use. It is a promising start though, showing that building a popularity predictor for BYU speeches through machine learning is a viable possibility. Additionally, the type of intelligibility that the decision tree gives is invaluable for members of the BYU speeches team hoping to make data-driven decisions for their company, intelligibility which is impossible with any of the other models tested.

Similar work has been requested on the university-sponsored magazine.byu.edu. This will be especially interesting, since this content is generated by members of the speeches team, not invited speakers. Thus, changes could be made in the content to optimize for success, based on the prediction of the model.

We also recognize some limitations with the features and labels which we have chosen, which have restricted the success that our model has been able to reach. We conjecture that a more accurate label could be obtained by looking only at the number of page views for the first 6 months since a speech was posted to the website. We did not include this label in our research since if the speech was delivered a long time ago but only recently posted to the speeches website, then it is unlikely that the post date will be recorded anywhere. We believe that the labels that we chose optimized both for accuracy and for volume of speeches that we could analyze, but further work could build upon these labels.

To improve our predictors, we have considered adding several features. For example, we have a Python script that determines if a topic is a significant portion of a talk. We could find which of these topics are the most and least popular and then add these as features to our model. We have also considered repeating our results for several other machine learning models, to see if any of them produce better results. Additionally, we could bin the page views into a larger number of bins (e.g. unpopular, semi-unpopular, neutral, semi-popular, and popular), and see how our models perform.

6. Acknowledgments

Thank you to BYU speeches for giving us access to the Google Analytics data for their website.

7. References

1. R. Bandari, S. Asur, and B. A. Huberman. The pulse of news in social media: Forecasting popularity, 2012.

2. X. Zhang, X. Chen, Y. Chen, S. Wang, A. Li, and J. Xia. Event detection and popularity prediction in microblogging. Neurocomputing, 149:1469–1480, 2015.

3. "Social Security." *Social Security History*, Social Security Administration, www.ssa.gov/oact/babynames/limits.html.

4. "API Reference." *Contributing Guidelines - TextBlob 0.15.1 Documentation*, textblob.readthedocs.io/en/dev/api_reference.html.

8. Endnotes

1 Weka's wrapper algorithms only support F1 score, not the more general Fß score. We modified Weka's source code so that we could run the wrapper with other F scores; however, this ultimately didn't improve the models' F0.2 score when compared to subsets found using the F1 score. Thus, we decided to use F1 score for feature selection even though we use F0.2 score for final evaluation.