UNIVERSITY OF TEXAS AT AUSTIN

OPERATIONS RESEARCH AND INDUSTRIAL ENGINEERING

Text Classification from Labeled Documents

Author :
Deepak mahapatra

Advisor :
Dr. Matthew Hersh

SUMMARY

I. Abstract

Data exploration and text mining have become very popularly used tool for decision making applications in the recent times. One of the interesting and useful part in this domain is the use of text data for generating features which can help in solving many supervised learning models. This project used K means clustering and latent semantic analysis (LSA) for dimensionality reduction of the term frequency-inverse document frequency (tf-idf) matrix generated from the text data and then compared the accuracy of classifications using Extreme Gradient Boosting (XgBoost) with other classification algorithms. An example data-set containing various features of websites including raw text in the web-page,to find whether the website is evergreen or not, was used to validate the methodology adopted in this project. The accuracy of classifications and the area under ROC curve was higher when using the tfidf scores with reduced components using LSA as compared to the the K-means clustering. The findings can be used for classification models used for recommendation systems,product review classifications and twitter, facebook post analysis.

II. Introduction

A. *Text Feature Extraction*

Text data has been increasingly used for various machine learning models. In the field of product reviews, text book data, web logs, transcripts and news articles there are great scope of applying algorithms to create meaningful information. However to achieve that the text has to be presented in a manner of numerical values which can be used for model building. All the documents should be of the same size to run an uniform model. In order to achieve this the various steps required are

- Tokenizing: Each document need to be converted to an array of words. Each word in the string is given an integer id.
- Stop word removal: The words which do not add any meaning to the text not in form meaning instead of grammar correction are removed. Examples are articles a, an and the.
- Stemming:Equal meaning words are converted to one word. Examples are plural form, different tenses of a single word
- Frequency: Counting the occurrences of each word in the documents
- Normalizing: Normalizing the importance of words in the corpus.

After generating the frequency of all the words the filtered word which have a significant importance in terms of frequency are treated as the features. Each document is converted to a matrix of the words with the normalized frequency of each word in that document.

B. *Tf-idf Vectorizer*

Whenever there is a large corpus the high frequency words may have a lesser significance than some low frequency words which may carry meaningful information. So by using the frequencies directly the high frequency words may dominate the classifier model. To avoid these occurrences the Tf-idf vectorizer is used where the term Tf means term-frequency while the term tfidf means term-frequency times inverse document-frequency:

$$tf - idf(t,d) = tf(t,d) \times idf(t)$$

Where inverse document frequency (idf) is given by

$$idf(t) = log\frac{1+N_d}{1+df(d,t)} + 1$$

C. *Latent Semantic Analysis*

Latent Semantic Analysis is a variant of singular value decomposition (SVD) that only computes the k user defined largest singular values. When truncated SVD is applied to term-document matrices or the tf-idf matrix it is known as latent semantic analysis (LSA). LSA is in particular beneficial for term-document matrices which are overly sparse. The corpus is a matrix of size $t \times d$ where t is the terms and d is the documents. Using singular value decomposition this matrix is represented as a product of three matrices as $X = T \times S \times P^T$ which is the SVD [4]. Here T is a $t \times r$ matrix, P is a $p \times r$ matrix and S is a $r \times r$ matrix .S matrix consists of the singular values and T corresponds to the term vector and D corresponds to the document vector which is also a type of representation of the matrix using orthogonal indexing dimensions. Using a truncated SVD, LSA keeps only the k largest singular values and the associated vectors.

D. *K Means Classifier*

Data clustering is used to find natural groupings in a set of observations for a very large number of variables.The main purposes solved by data clustering is

- To gain insight into the data
- To identify similarity between the variables
- To minimize the size of the observations

K means clustering is a method used in data mining for partitioning n observations into k clusters where each observation is nearest to one of the k means. Let X , $i = \{1; ...; n\}$ be the set of n dimensional points. And to form k clusters and partition these data points to one of these cluster, $C = \{c_k; k = 1; ...; K\}$.

The K means algorithm tries to minimize the square error between the empirical mean of the clusters and the points in the cluster [2].

$$\min \sum_{k=1}^{K} \sum_{X_i \in C_k} \| (X_i - \mu_k) \|^2$$

The procedure followed by the algorithms is

- initialize partition of K random clusters
- follow the next two steps until the cluster member stabilizes
- Generate new partitions by assigning each pattern to its closest cluster center
- Find the new center of the clusters.

E. *Extreme Gradient Boosting*

Extreme gradient boosting is becoming an increasingly popular method used when processing a large amount of data to train a model. It is one of the most promising boosting algorithm which has a high scalability. Sparse data are handled by a novel tree learning algorithm. A weighted quantile sketch procedure handles the instance weights in approximate tree learning [5]

III. Methodology

In this project in order to verify the results I have used the StumbleUpon website database from Kaggle. The description of the data set is given in table I as was provided by the data owner [1].First I have applied lemmatization on each documents from the database. The tfidf vectorizer of sklearn library in python was applied to convert each frequency matrix to a normalized matrix.

The steps I follow in our modelling are as follows:

- Create new documents from the text data of each observation
- Pre-process the text as in Section 2.1.

FieldName	Type	Description
url	string	Url of the webpage to be classified
urlid	integer	StumbleUpon's unique identifier for each url
boilerplate	json	Boilerplate text
alchemy_category	string	Alchemy category (per the publicly available Alchemy API found at www.alchemyapi.com)
alchemy_category_score	double	Alchemy category score (per the publicly available Alchemy API found at www.alchemyapi.com)
avglinksize	double	Average number of words in each link
commonLinkRatio_1	double	# of links sharing at least 1 word with 1 other links / # of links
commonLinkRatio_2	double	# of links sharing at least 1 word with 2 other links / # of links
commonLinkRatio_3	double	# of links sharing at least 1 word with 3 other links / # of links
commonLinkRatio_4	double	# of links sharing at least 1 word with 4 other links / # of links
compression_ratio	double	Compression achieved on this page via gzip (measure of redundancy)
embed_ratio	double	Count of number of >usage
frameBased	integer (0 or 1)	A page is frame-based (1) if it has no body markup but have a frameset markup
frameTagRatio	double	Ratio of iframe markups over total number of markups
hasDomainLink	integer (0 or 1)	True (1) if it contains an >with an url with domain
html_ratio	double	Ratio of tags vs text in the page
image_ratio	double	Ratio of >tags vs text in the page
is_news	integer (0 or 1)	True (1) if StumbleUpon's news classifier determines that this webpage is news
lengthyLinkDomain	integer (0 or 1)	True (1) if at least 3 >'s text contains more than 30 alphanumeric characters
linkwordscore	double	Percentage of words on the page that are in hyperlink's text
news_front_page	integer (0 or 1)	True (1) if StumbleUpon's news classifier determines that this webpage is front-page news
non_markup_alphanum_characters	integer	Page's text's number of alphanumeric characters
numberOfLinks	integer	Number of >markups
numwords_in_url	double	Number of words in url
parametrizedLinkRatio	double	A link is parametrized if it's url contains parameters or has an attached onClick event
spelling_errors_ratio	double	Ratio of words not found in wiki (considered to be a spelling mistake)
label	integer (0 or 1)	User-determined label. Either evergreen (1) or non-evergreen (0); available for train.tsv only

TABLE I: The description of data used
Note. Data description is used as provided by Kaggle [1].

- Create document vs word count matrix
- Convert these document vs word matrix to generate the tf-idf sparse matrix.
- Apply Latent Semantic Analysis using the Sklearn TruncatedSVD library
- Apply K means clustering on the tfidf matrix generated in the previous step.
- Apply Xgboost, Logistic Regression and Random Forest on the complete data-set using the features generated in the previous step.
- Generate ROC curve for each combination of algorithms to find the best suited algorithm.
- From the ROC curve generate a threshold level for true positive rate and false positive rate to build threshold for the classifier models and build the confusion matrix.

IV. Results

In this section I will discuss the results that were obtained by working on the StumbleUpon data-set. StumbleUpon is a web content discovery engine, that was developed by the inputs of user on websites, that gives recommendation for relevant, high quality pages to its users, based on the user preferences. The websites they recommend can either be classified as ”ephemeral” or ”evergreen”. So I built a classifier model using this data set is to use the text data from each of the website they have in their database along with other attributes for predicting the evergreen or ephemeral state of a website.

The data contains 26 factors for each of the websites. The missing data in all of the continuous numeric variables type were replaced by the mean of all the observations of that variable and the categorical were replaced by the majority category of that variable, except for alchemy category in which the missing data were categorized into the unknown category. After completing this pre-processing steps I have applied the tf-idf vectorizer on the text data to generate new features. This tf-idf was then converted into clusters by K means clustering methods and used with the other variables in the data set to build a classification model using Extreme gradient boosting, Logistic Regression and Random Forest algorithms. The results obtained with these models were compared with a model which used the reduced matrix by applying LSA instead of K means on the tf-idf matrix generated from the text data.

In order to evaluate the best model among all the generated models I used ROC curve or lift chart as shown in figure 1 which plots the true positive rate to the false positive rate in the classifications generated by the models. The worst case scenario being the case when I randomly assign classes to the factor variables being the base line with 45 degree slope. The curve which covers the maximum possible area in the figure is treated as the best model. In my case the logistic regression model with the text components generated by Latent Semantic Analysis performed better as compare to the other models. However, the gradient boosting method produced nearly similar values of area under curve with training time significantly lower than the other methods.

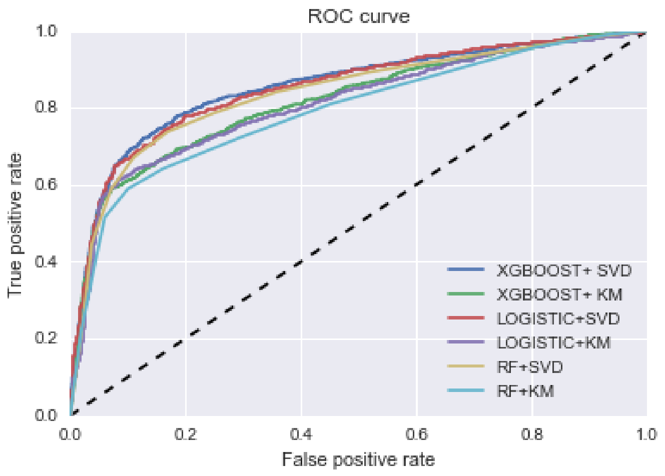

Fig. 1: ROC curve for different models

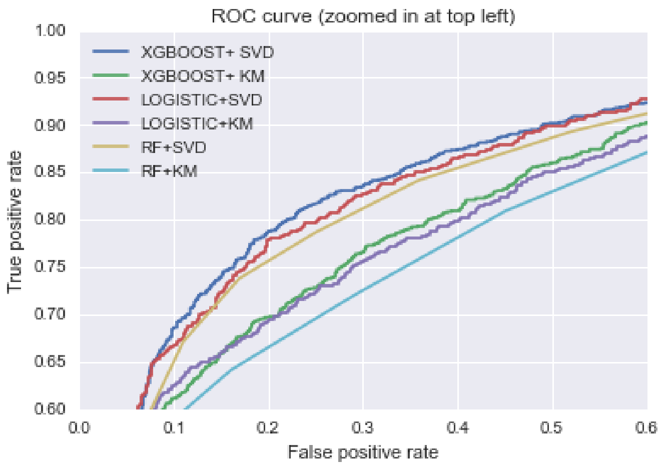

Fig. 2: ROC curve for different models with zoomed in on 0.6 true positive rate and above

So in order to further see the performance of the LSA I analyzed each of the columns generated by the tf-idf vectorizer and the percentage of variance explained by each of the column. As it can be seen from the graph in figure3, 80 percent of the variance was explained when I reduced them to 100 number of individual components in LSA. So I used 100 as the number of reduced columns in the tf-idf matrix and used these for the classification model.

As it can be seen from the figure 4, there is a correlation between some of the variables. However when using the factors in the classification models the significance of these correlated factors is eliminated by the algorithms. The feature importance score also gives a clear indication of removal of these variables from the model.

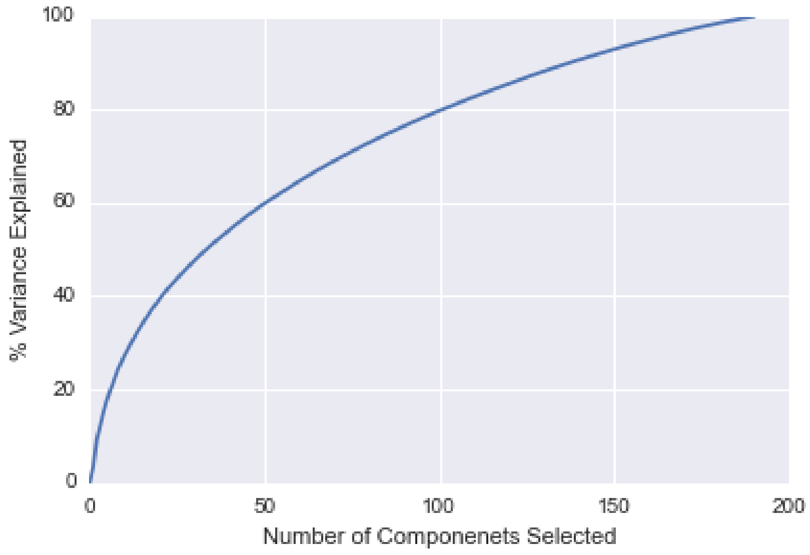

Fig. 3: Percentage variance explained vs the number of reduced components

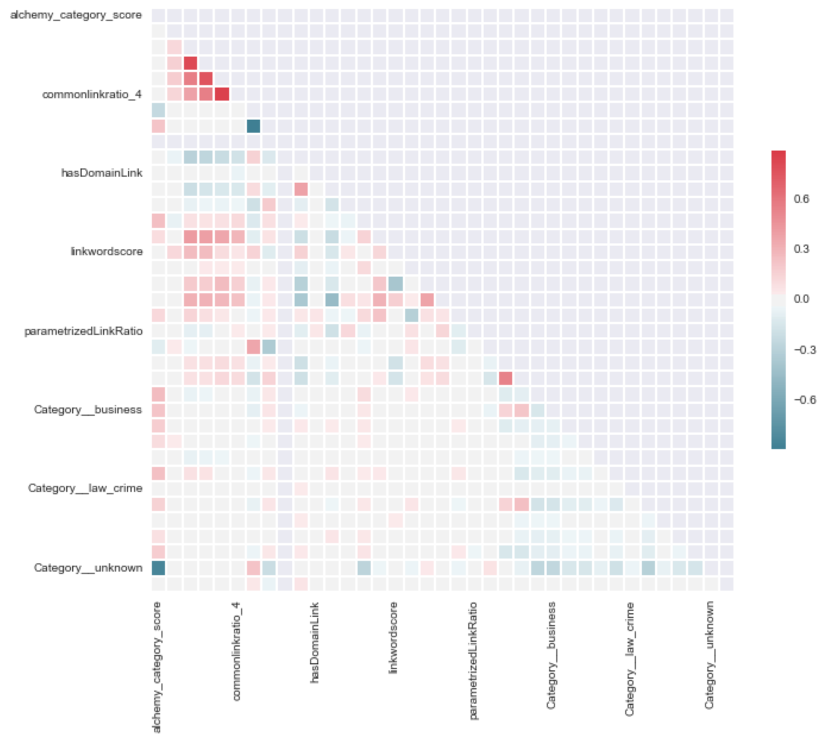

Fig. 4: Correlation among the predictor variables excluding the text data

The figure 5 gives the feature importance of each of the variables when K means clustering is used to generate clusters bases on the tf-idf matrix. The number of clusters generated in our case is 2 as I have only two classes of the predicted variable. However this cluster can be changed in order to check the performance of the algorithm. As evident from the results here the F score of the clusters generated by K means clustering is not among the best predictor variables.

When I plotted the feature importance of each of the reduced component of LSA as in figure 6, I found that each of these column has a significant F score in predicting the classes. So these values can be used in the model in order to get a better accuracy in the predictions. This also gives us a confirmation on the

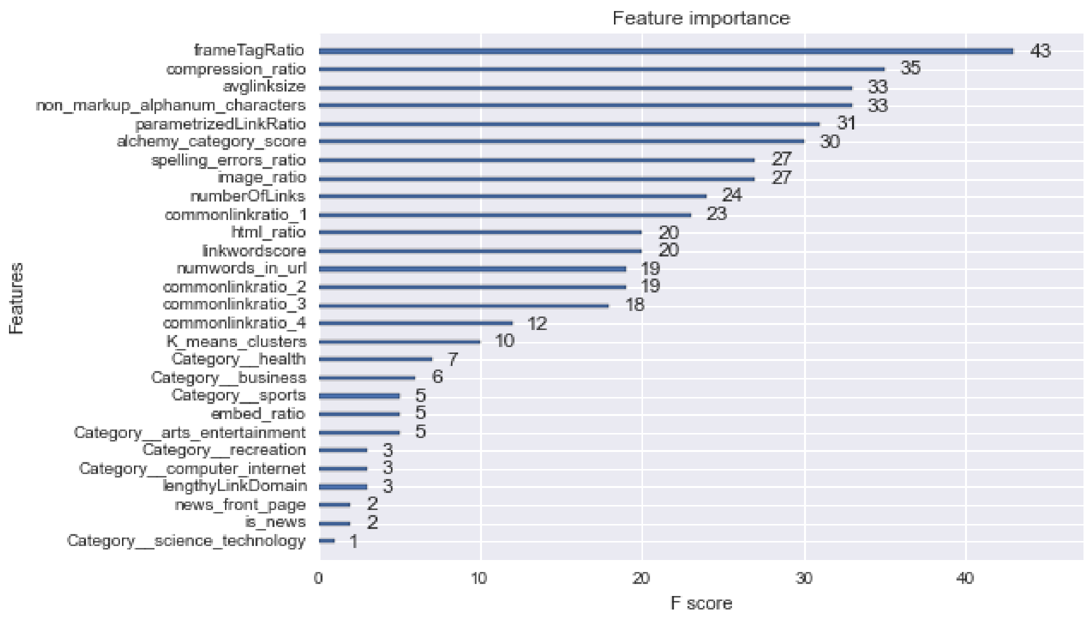

Fig. 5: Feature Importance

selection of number of reduced components being adequate in our model.

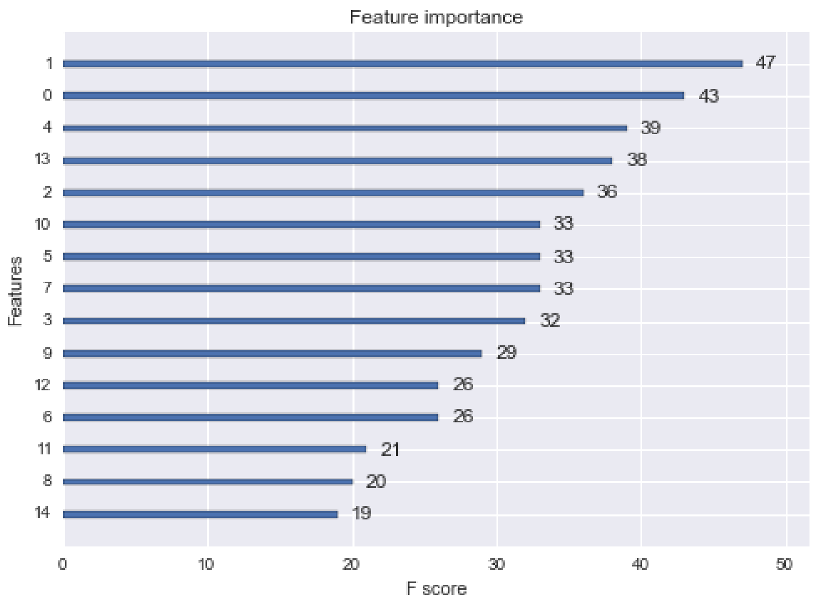

Fig. 6: Feature importance using the reduced components only of LSA for prediction

As it can be seen from the graph of feature importance in figure 7, the individual components of the reduced tf-idf matrix has significant F scores with ranking in the top 10 predictor variable range which is a clear indicator that some variability in the target variable was lost when I used K means clustering or any clustering method to reduce the tf-idf matrix. This also gives an indication on which of the columns are significant in predicting the result so the other columns in the reduced matrix can be ignored and thus reducing the computational time.

As evident from the graph in figure 3, I chose a cutoff value for the predictions from a probability of around 0.7 in all the models to calculate the confusion matrix values. The true positive, false positive, true negative and false negative predictions of each of the model I built are given in table II.

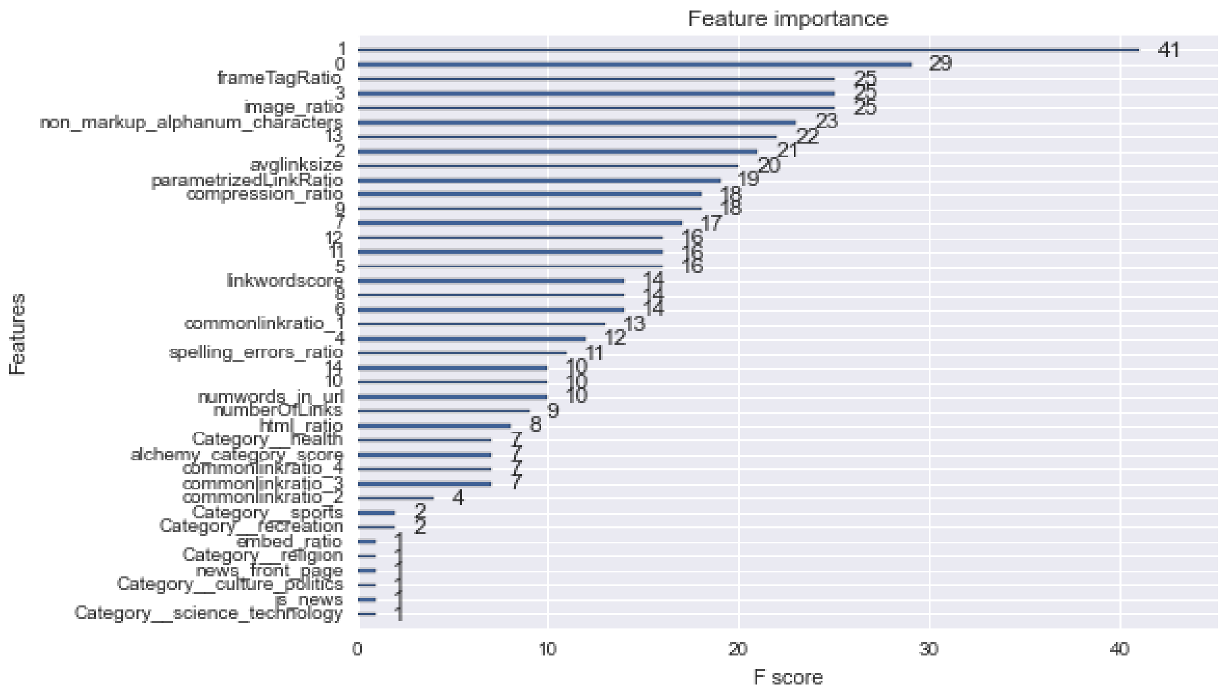

Fig. 7: Feature importance when using the reduced components of LSA along with other variables for prediction

Model	True Positive	False Positive	False Negative	True Negative
KMeans + XgBoost	1036	60	499	623
LSA + XgBoost	1036	60	499	623
LSA + Log reg	934	162	332	790
K means + Log Reg	984	112	429	693
LSA + Random Forest	920	176	362	760
K means + Random Forest	882	214	398	724

TABLE II: Confusion Matrix for all the models

Model	Accuracy	Precision	Recall	Area Under Curve
KMeans + XgBoost	0.7480	0.7949	0.7479	0.8128
LSA + XgBoost	0.7755	**0.7979**	**0.7660**	**0.8411**
LSA + Log reg	**0.7773**	0.7843	0.7773	0.8360
K means + Log reg	0.7561	0.7796	0.7561	0.8092
LSA + Random Forest	0.7574	0.7653	0.7574	0.8162
K means + Random Forest	0.7241	0.7309	0.7241	0.7893

TABLE III: Accuracy scores among models

V. Conclusion

The results in this project led to the conclusion that when a larger data set of text data and a higher dimension in the tf-idf matrix generated are used ,Truncated SVD or LSA provides a better reduction in the dimension as compared to the clustering methods such as K means clustering. Also the classification accuracy by the extreme gradient boosting is higher as compared to the classification accuracy achieved by the random forest or logistic regression methods. The training time in Xgboost being small as compared to other classification methods, makes it a very efficient algorithm to use when a large data set is used, especially text data where the training time of the model is a significant factor for model selection.

References

[1] https://www.kaggle.com/c/stumbleupon/data

[2] Hartigan, John A., and Manchek A. Wong. ”Algorithm AS 136: A k-means clustering algorithm.” Journal of the Royal Statistical Society. Series C (Applied Statistics) 28.1 (1979): 100-108.

[3] Landauer, Thomas K., and Susan Dumais. ”Latent semantic analysis.” Scholarpedia 3.11 (2008): 4356.

[4] MLA Landauer, Thomas K., Peter W. Foltz, and Darrell Laham. ”An introduction to latent semantic analysis.” Discourse processes 25.2-3 (1998): 259-284.

[5] Chen, Tianqi, and Carlos Guestrin. ”Xgboost: A scalable tree boosting system.” Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016.