CIS 787 Final Project Report
Akash Rai, 919165908

Web Link categorizer

1. Introduction

In today's world of texting, we receive a lot of text messages containing web links on messaging platforms like WhatsApp, Facebook Messenger, and iMessage. Sometimes it becomes tedious and boring to open and read out every link that friends or the general public send out to you. The feature of having information about the link beforehand becomes very valuable. Possessing knowledge about the URL can assist in deciding whether to open the link or to ignore it. In this project, I tried to develop a model to classify URLs into five categories such as Sports, Politics, Technology, Business, and Health.
Usually, URLs are very expressive and describe the general idea about the content on the page they point. And in this age and time of search optimization, including keywords in the URL is one of the criteria for getting good results in the search ranking. Consider a URL, https://www.cricbuzz.com/cricket-news. This URL contains keywords like https, www, cricbuzz, cricket, and news and these can be used to classify the URL into a category like sports. Many papers in this domain ignore the URLs and focus principally on the content they have. But in this project, I've attempted primarily to focus on the URLs and title of the page. Hence the model training time decreases significantly and could achieve up to 88% f1-measure in the process.

2. Prior work

The author in the paper [1] has described a brief summary of how the URLs can be used for classification purpose. He has outlined a simple model to classify URLs and included quantifying the results using the content from the web page. The ideas discussed in the paper [1] have encouraged my primitive experiments. Paper [2] explains methods to perform Latent Semantic Analysis (LSA) using SVD to summarize the webpage. I find it a very interesting idea to summarize URLs which can specify more details to the users.

3. Model

3.1 Dataset

The dataset for this project consists of URLs, title on the page of the URL and the category it belongs. I accumulated approximately 100k URLs personally using Google search. In the general cleaning process, I removed the duplicates and those URLs which were cross-listed among various categories. After the initial cleansing process, I ended up with approximately 80k unique URLs. The general structure of URL is described as <scheme>://<host>/<keywords>/<filename>. So, I have exploited the fact that many URLs contain some keywords describing the page they are pointing.

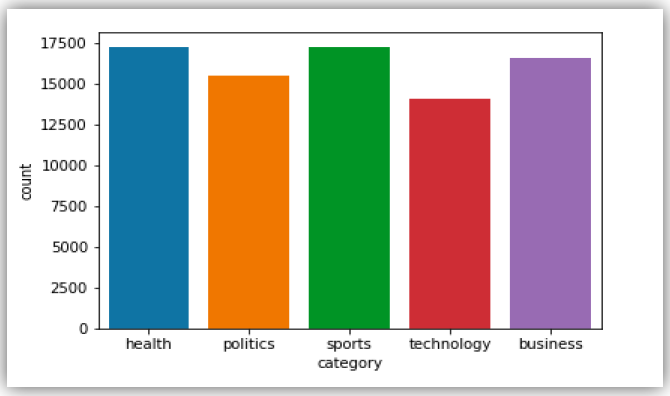

Figure 1: Dataset Information

3.2 Preprocessing

In the preprocessing process, firstly, I extracted the single word tokens from the URLs and its corresponding title. Then discarded the special characters and numeric values from the alphanumeric words. The URL https://www.politico.com/story/2018/12/06/pelosi-schumer-trump-wall-funding1046443 after the preprocessing step would get transformed into a set of tokens {https, www, politico, com, story, pelosi, schumer, trump, wall, funding}. Next is a process of eliminating stop words from the set of selected tokens. After removing stop words the new set becomes {politico, story, pelosi, schumer, trump, wall, funding}. The same process is again followed for the title. Finally, we end up with a set of tokens for each data entry in the dataset.

3.3 Feature Creation

Many a time, URLs contain keywords which make more(different) sense when we see it in a different context. To capture the context, we must consider them in a group. Therefore, I have used a shingling process to generate a set of k shingles which help to capture the context of the keywords. I performed an analysis to find the optimal number of k-shingles. The result (Fig. 2) showed that using up to 6 shingles provide good accuracy. Finally, a comprehensive set of k-shingles (1-6) is generated using the set obtained in the preprocessing step.

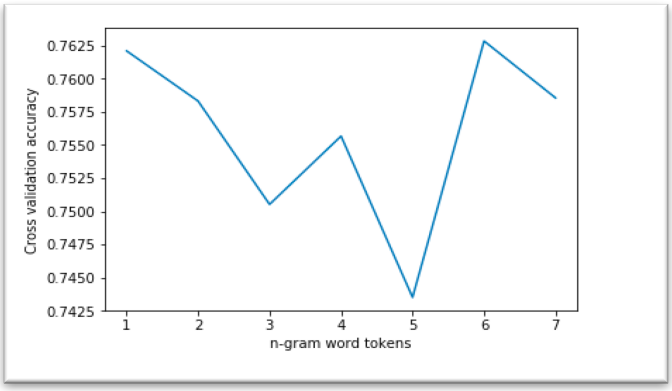

Figure 2: n-grams analysis

Once we have a large set of tokens, using the vectorization process we generate our features using Tf-IDF (or Count) vectorization technique.

3.4 Feature Selection

The feature selection is important in the sense that not all the features contribute equally. Many of them have a little contribution and hence can be discarded with a little trade-off in accuracy. Thus, this reduces the number of dimensions a data point has and in turn reduces the training time of the model. I experimented with various feature selection methods such as chi2, mutual information, variance thresholding, recursive forward elimination, PCA and SVD [2]. All the mentioned techniques had reasonable performance compared to each other in terms of accuracy. Accordingly, I proceeded with the chi2 technique which has a decent tradeoff between processing time and accuracy.

The number of features created following the vectorization process is high but the number of features that actually contribute is far less. Therefore, I performed an analysis to find the number of features that must be selected finally to train the classifier. The results (Fig. 3) suggest selecting top 2200 features from the feature set. Feature selection step drastically reduces the number of features from 90k to 2.2k top contributing features.

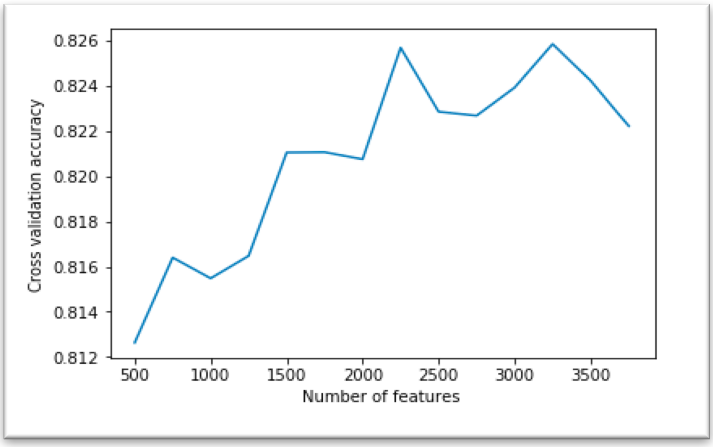

Figure 3: Analysis for Number of features

Dimensionality reduction techniques such as Principal Component Analysis (PCA) and Singular value decomposition (SVD) were explored. PCA works by finding principal components, which explain the most variance in the data. Top principal components can thus be used to reduce the feature set. SVD works by finding singular concepts, and the dimensionality can be reduced by ignoring the lower singular values. Moreover, we can use SVD to perform Latent semantic analysis [2] to find concepts which can be recommended to users along with the category. In the graph (Fig. 4), we can see that for PCA and SVD, as the number of components increase, the accuracy decreases. Moreover, both PCA and SVD techniques work in an unsupervised manner whereas techniques such as chi2 work in a supervised manner. Thus, PCA may remove features which are important for classification. Therefore, I finalized our feature selection process using the chi2 method.

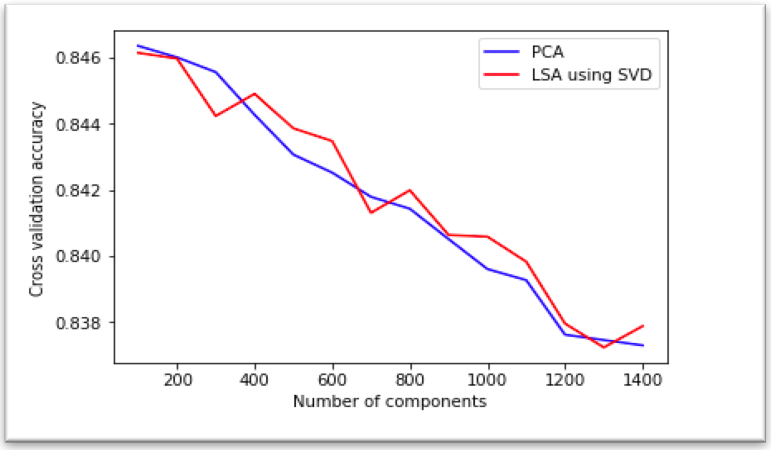

Figure 4: Number of components using PCA and SVD

A word cloud (Fig. 5) of top 100 words is generated after feature selection. We can see that the words like {election, diet, market, football, tech, Olympic, entrepreneur} get captured and these words have some resemblance with categories they belong.

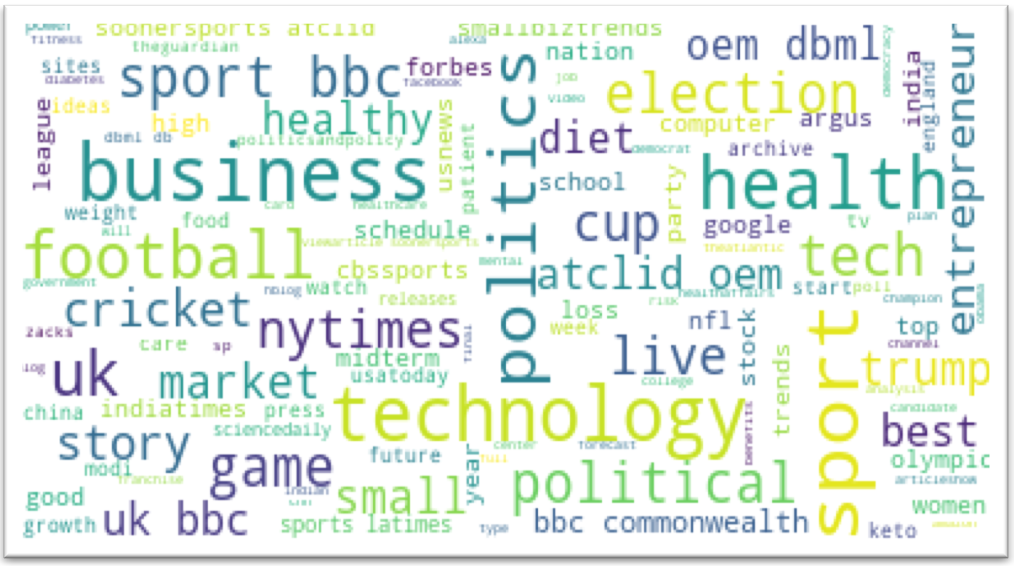

Figure 5: Word cloud generated from top 100 words

3.5 Classifiers

The resulting set of features are used to train the classifier by splitting the dataset in 75:25 ratio. 75% of the data entries are fed to the classifier and remaining 25% is reserved for cross-validation testing purpose. Multiclass classification experiments were performed using multiple classifiers, such as Decision tree, Naïve Bayesian, K-Nearest Neighbors, Random Forest, Logistic regression and Support Vector Machines (SVM) [1]. The accuracy of the classifiers is somewhat similar.

a. Naïve Bayesian classifier

The Naïve Bayesian (NB) classifier is a simple and effective classification algorithm. The basic idea in NB classifier is to use the joint probabilities of features and categories to estimate the probabilities given a document.

b. Decision tree classifier and Random Forest classifier

Decision tree classifier works by building a set of nodes by splitting based on some features. Leaves of the decision tree indicate the class of the instances. Predictions are performed by following the path in the decision tree. Random forest is an ensemble of multiple decision tree classifiers. Each classifier training on a different sample of a dataset. The analysis for depth of the tree (Fig. 6) was performed, and we can see that after depth 60, the accuracy starts to saturate. Further increasing depth might introduce overfitting. Therefore, I picked the max depth 60 for the decision trees.

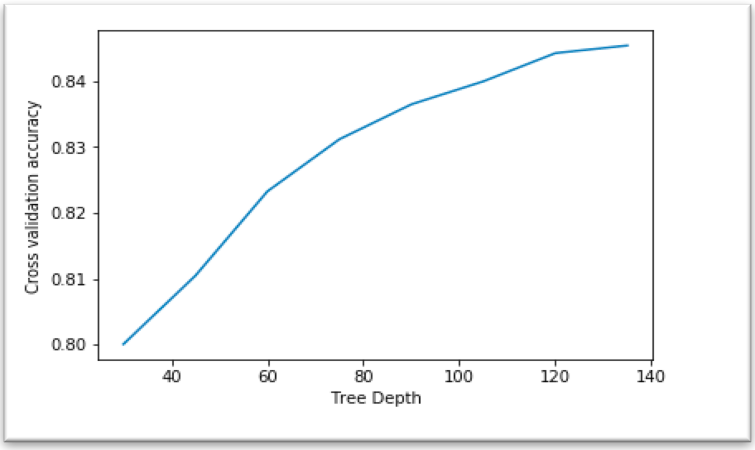

Figure 6: Maximum tree depth analysis

c. Support vector machines

Support vector machines (SVM) are a supervised learning model which tries to map a hyperplane in the space which divides the data instances as wide as possible. In the prediction stage, the new points are mapped in the same space and predicted based on the side of the hyperplane.

4. Experimentation and Result

Random 25% of the dataset was sampled for the model evaluation purpose. The classifier then predicts the categories of the test set. I've have employed standard precision, recall and F1-measures to evaluate the model. Precision (p) can be described as the proportion of actual positive class instances predicted by the classifier among all the positive predicted class instances.

Recall (r) can be described as a proportion of predicted positive instances among all the actual positive class instances.

F1-measure is the harmonic average of precision and recall.

F1-measure = 2pr / (p+r)

The experimental setup included ~ 60k instances in the training set and ~ 20k instances in the test set. The results are described in the following table.

Classifier	Number of features	Training time (in sec)	Prediction time (in sec)	Accuracy (%)	F1-measure (%)
Decision Tree (Depth 60)	2200	88.51	0.12	76.19	78
	2500	99.72	0.15	76.3	78
Naïve Bayes	2200	0.37	0.08	85.21	85
	2500	0.37	0.06	85.23	85
KNN (N = 50)	2200	-	-	-	-
	2500	30.47	3794.31	81.13	81
Random Forest	2200	88.72	0.60	83.12	84
	2500	103.38	0.68	83.27	84
Logistic Regression	2200	25.13	0.05	88.22	88
	2500	29.29	0.06	88.36	88
SVM (Linear)	2200	2.15	0.06	88.50	89
	2500	4.74	0.10	88.69	89

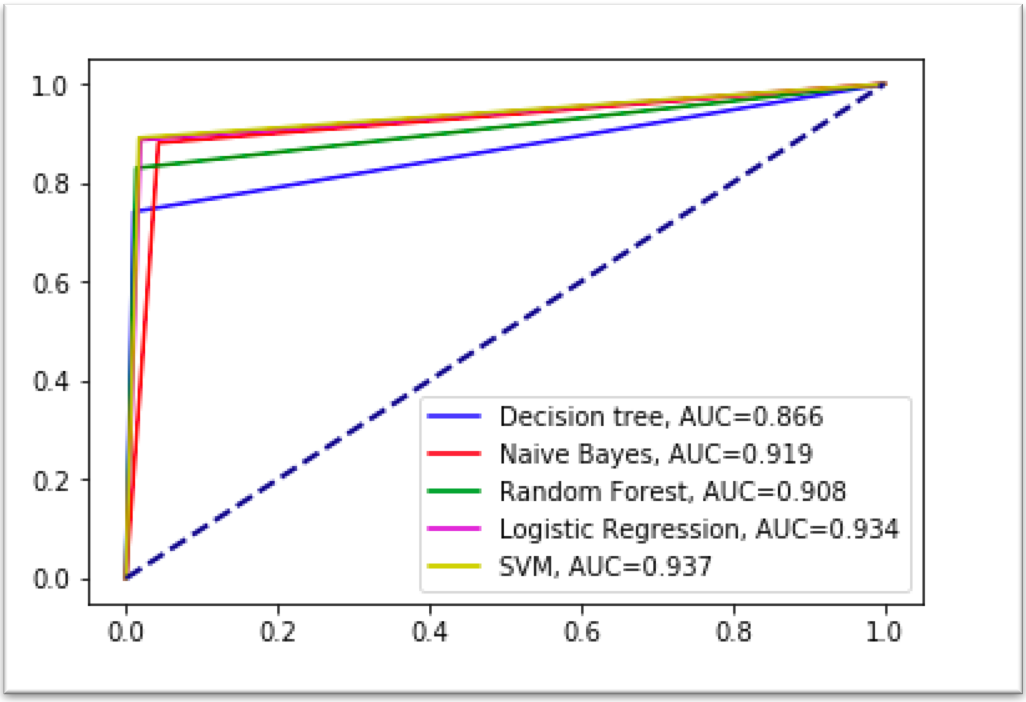

Figure 7: ROC curves of classifiers

From the experiments and the ROC curve (Fig. 7) we can observe that the Naïve Bayes, Logistic Regression and SVM have comparable accuracies having inexpensive training and testing time. Therefore, we can have any one of those as our final classifier.

5. Conclusion

With the series of experiments, we conclude that we can achieve sensible performance in a remarkably short time to assist users to classify the URLs into various categories. The URLs can be categorized on the fly and displayed on the messaging platforms. This helps users to decide

whether to read the content on the URL or to ignore it. By no means, this model can predict every URL correctly, but my work in this project indicates that we can build better models by further analyzing new methods and experimentations to come up with a refined multiclass model.

6. Future work

Lemmatization and stemming can be applied to transform tokens into their root words. This may increase the accuracy of the model. As words like playing, played can be converted to play. This reduces the feature set to a certain extent.
Use of text from the URL page can enhance the model by providing more detail about the topic. But it will increase the analysis and training time. Thus, we would have to analyze the complexity of adding larger texts in the model. Also, we could use another model having Ensemble classifiers classifying URLs and the titles separately.
Finally, I could not completely develop the feature to summarize URLs using LSA. The results were not as per expectations. The work from this project can be extended to develop a URL summarizer.

7. References

[1] Min-Yen Kan. 2004. Web page classification without the web page. In Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters (WWW Alt. '04). ACM, New York, NY, USA, 262-263. DOI: https://doi.org/10.1145/1013367.1013426
[2] Dou Shen, Zheng Chen, Qiang Yang, Hua-Jun Zeng, Benyu Zhang, Yuchang Lu, and Wei-Ying Ma. 2004. Web-page classification through summarization. In Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '04). ACM, New York, NY, USA, 242-249. DOI: https://doi.org/10.1145/1008992.1009035