Classifying Ephemeral vs Evergreen Content on the Web

29.04.2017

Gurjot Singh Walia (140020121)
Abhinav Rondi (140050054)
B Srinivas Naik (140050064)
Goutham Arukonda (140050065)

Overview

Web classification is a very important machine learning problem with wide applicability in tasks such as news classification, content prioritization, focused crawling and sentiment analysis of web content.

With the rapid proliferation of user-generated content available on the Internet, one of the biggest challenges is determining the relevancy of the information shown. Many sites present today implement powerful recommender engines to sort the content present in a website into different categories. These engines learn and are trained from behaviour of different kind of users and their interaction with the content of the site.We can divide the web content information in two broad classes:

1. Newsy, ephemeral: This class contains documents that are time sensitive and ephemeral. Typical examples are news articles, classified listings, blog posts about current or local events. Typically these documents receive a big, short lived traffic spike. For example, a sports analyst may write an article about possible outcome of a game, which becomes almost instantly obsolete as the game concludes.

2. Evergreen: These are documents that endure the test of time. Typical examples are Well-written, informative articles that tell the users how something works, High quality humor or opinion pieces, Literature, arts and entertainment articles. These can be truly timeless.

A robust evergreen content classifier can alleviate this content discovery problem. The challenge we face in this project is classifying the contents in a website into one of these categories mentioned. This information is valuable piece of cake for web advertising industries as they can use this information as basis to decide the place and time to show their advertisements on the website, to catch up right people at right time. This information can also be used for content sites interested in capacity planning for hosting different pages based on expected longevity.

Goals

1. Learning and exploring many new machine learning classification algorithms.
2. Getting exposed to various interesting features and components of webpages used in predicting user's interests which form major part of modern day advertising market.

Data Set Description

StumbleUpon and Kaggle provided a training set containing 7395 web documents, and a test set with hidden labels containing 3171 documents. Following is a summary of features available in this data set.

- **Text content features:** Full HTML source of the pages.
- **Document structural features:** Ratios of tags vs text, image vs text, spelling error counts, URL word counts.
- **Derived features:** compressed document size (as an approximate indicator of redundancy), document category, number of outgoing links in the page, linkword score (number of document words that appear in links) etc.

We also further divided the training set to create cross validation and test sets to measure the performance of various classifiers.

Feature Selection

The basic information available for each training example is the URL of the page. Along with the URL itself, Kaggle provides a snapshot of the HTML retrieved from the URL in raw form. The first challenge is to transform the HTML page into features that can then be processed by classification algorithms.

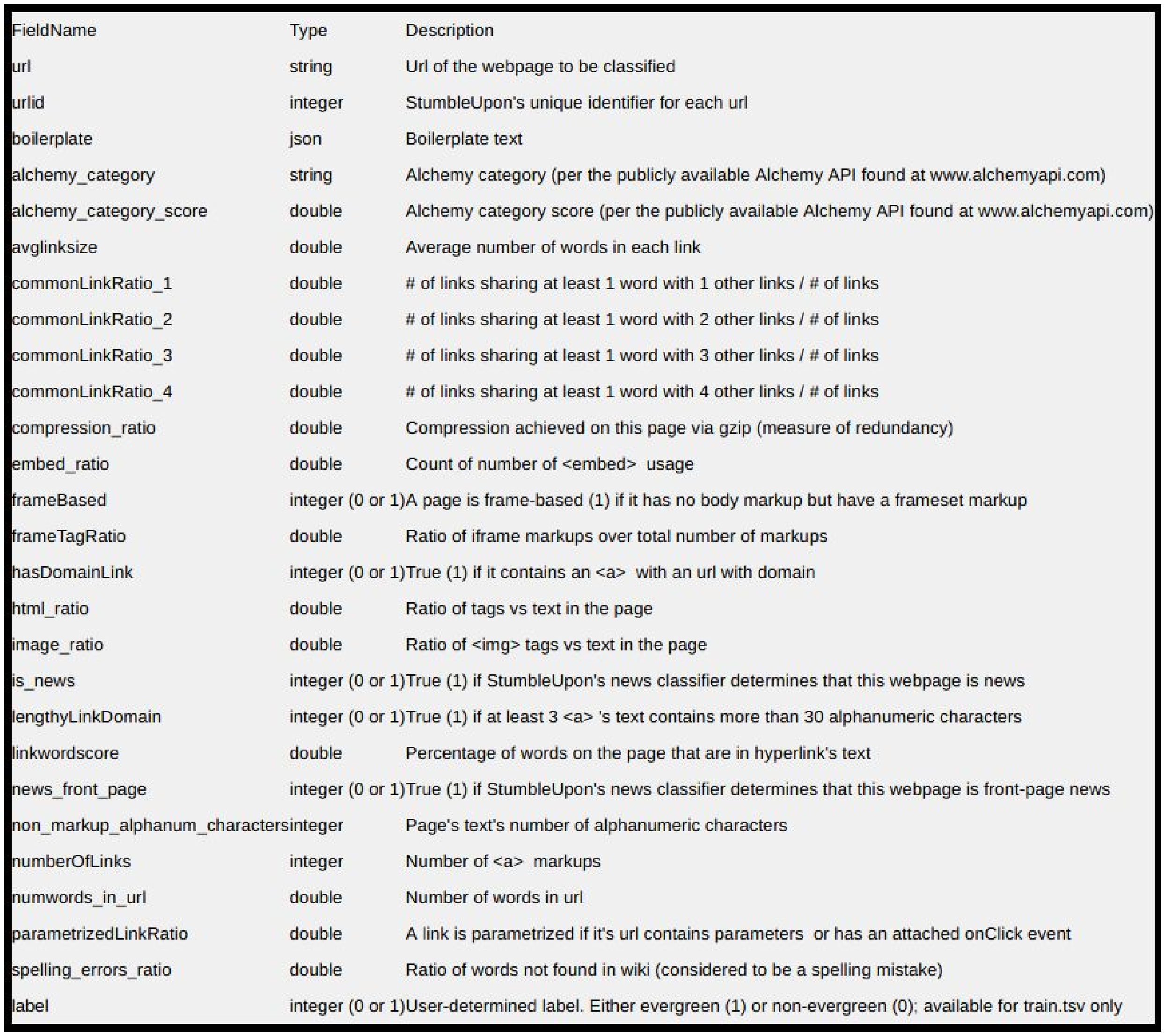
Some of the major features that were included in the classification based on their predictive possibilities are:

- **"URL":** Page URL
- **"Body":** Body text
- **"Outline":** Title and header node contents
- '**keywords**' metatag

A clear description of features extracted from the dataset is presented in next page.

The image below show the description of various features available in the dataset (this is extracted from kaggle competition page "https://www.kaggle.com/c/stumbleupon/data").

The following table includes field descriptions for train.tsv and test.tsv:



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 <embed> 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 <a> with an url with domain
html_ratio	double	Ratio of tags vs text in the page
image_ratio	double	Ratio of <img> 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 <a> '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 <a> 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

Preprocessing

The extracted features were preprocessed to transform them into feature vectors. First, the contents of each page were transformed to standard ASCII encoding. The body text, title and header node contents were common English-language words, and were stemmed and lemmatized using Porter stemmer and Wordnet Lemmatizer respectively.

- **Processing URL of the page:** This includes stripping digits, stop words and stripping words like 'http', 'https', 'www', 'com', 'net', 'org', 'm', 'html', 'htm'. The intermediate form obtained is then further processed by applying stemming algorithms to extract the domain from the URL.

- **Preprocessing page content:** This includes processing body and title in the boilerplate separately if either of them are actually present for example considered. Then this data is broken down to finer granularity of sentences and words to which stemming and lemmatizing algorithms are applied.

Lemmatization is closely related to stemming. The difference is that a stemmer operates on a single word without knowledge of the context, and therefore cannot discriminate between words which have different meanings depending on part of speech. However, stemmers are typically easier to implement and run faster, and the may have lower accuracy.

We computed the term frequencies of all words appearing in the entire training and test set, and discarded the most frequently appearing words by using some threshold on the term frequency. The rationale for this approach is to discard the filler words in the English language (such as “a” or “the”) which have high frequency but little information.

PRE-PROCESSING STEPS

1. **Tokenization:** Cut sequence of characters into sentences and then to word tokens
2. **Stemming:** Match words from the same root
3. **Stop Words:** Remove common words such as ‘a’, ‘and’, ‘the’

Binary Model:

This is the simplest model. We use binary values for the real valued function.

$$w(t,\ d) \text{ = (1 if word t appears in the document d, 0 otherwise)}$$

TF-IDF Model

We used the term frequency-inverse document frequency (tf-idf) of each word as a feature. The tf-idf is the product of the term frequency, indicating the number of times a word appears in a given document, and the inverse document frequency, which measures how commonly the word appears across all documents. The inverse document frequency is computed as

$$idf(t,\ D) = log \frac{|D|}{1 + |\{\, d \in D : t \in d\}|}$$

where D is the set of training examples (documents), |D| is the number of training examples, and |{d ∈ D : t ∈ d}| is the number of documents where the word t appears.

We used **TfidfVectorizer** function from **sklearn.feature_extraction.text** library which has a parameter *smooth_idf* (set to true) to smooth idf weights by adding constant '1' to the numerator and denominator of the idf as if an extra document was seen containing every term in the collection exactly once.

$$idf(t,\ D)\ =\ log\ \frac{1+|D|}{1+|\{\ d\in D : t\in d\}|}\ +1$$

Our representation of documents ignore the ordering of the words. For example, the phrase "Mary likes Cake" has the same representation as "Cake likes Mary". Research in information retrieval has shown that the ordering of words in a document has minor impact for many tasks, which leads us to adopt the tf-idf model.

Metrics

The metric used for evaluation is the area under the receiver operating characteristic curve (ROC AUC). The ROC is a characterization of the true positive rate against the false positive rate of a classifier. The true positive rate is also known as sensitivity, recall or probability of detection. The false positive rate is also known as fall-out or (1-specificity). Each prediction result or instance of a confusion matrix represents one point in ROC space. The area under the ROC curve is equal to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. So closer the value of roc-auc to 1, the better is our model.

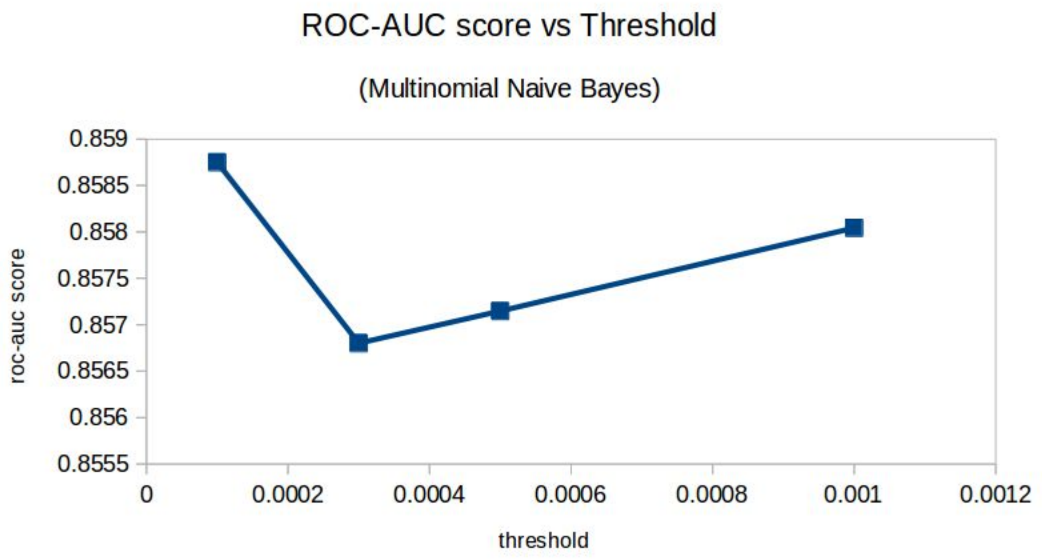
Classification and Results obtained

We applied three classification algorithms Multinomial Naive Bayes, Support Vector Machine & Regularized Logistic regression on the dataset. We began with the numeric metadata provided to get a better understanding of the problem at hand, but we iterated and improved on these models by adding the text features and other feature sets. We used 10 fold cross-validation in learning throughout this project unless mentioned otherwise.

Naive Bayesian:

We started by implementing the Multinomial Naïve Bayes using the boilerplate as the feature set as suggested using binary model. The average ROC_AUC score was 0.845. The first step in improving our model was to improve the feature selection. We started by giving equal weights to all tokens in Naive Bayes, which also included tokens not good for classification (stopwords).

We implemented mutual information based feature selection to filter out the stop words as mentioned in the class. It did show some improvement in filtering out most of the stop words, but several of the stop words didn't filter out. The key thing that was missing from mutual information metric is the importance of word frequency in a given document as well as the rarity of occurrence of that work. Hence we looked at TF-IDF (term frequency inverse document frequency). We performed feature selection using TF-IDF and then applied that to the Multinomial Naïve Bayes classifier. Below figure shows the variation of roc_auc score for different thresholds on fraction of frequency of occurrence of words in dataset.

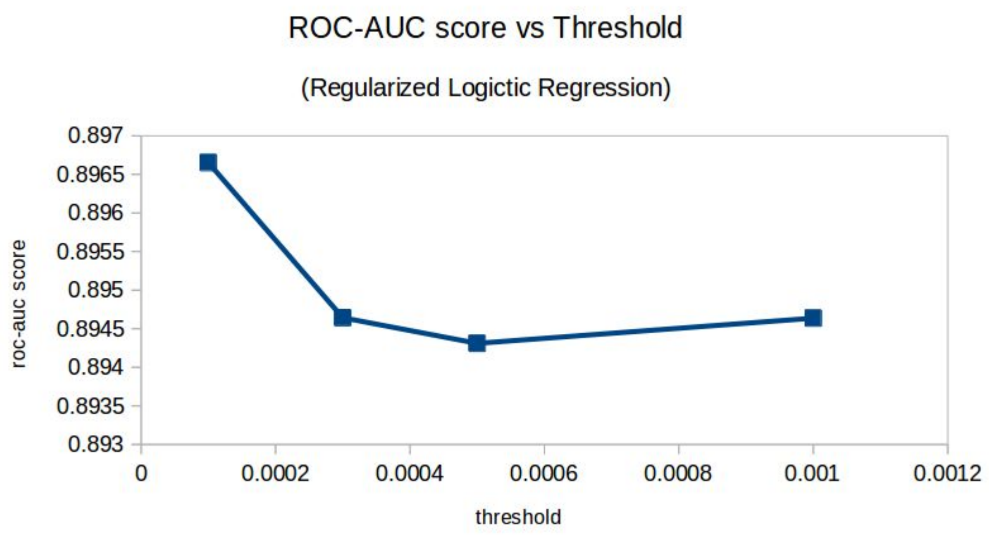


After finalizing the feature selection methodology, our next step was to start analyzing ways to improve the model. We reviewed the terms in boilerplate of the samples that were classified as false negative. We found and analyzed the following types of information: Empty body, Advertisements, Tabloid news, Bogus police report . Having an empty body or a tabloid news labeled as an evergreen clearly shows that there are human errors introduced during the manual classification. Given the noise in the dataset, we need to inspect models that perform regularization, like SVM or regularized logistic regression.

Regularized Logistic Regression:

From the Naive Bayes exercise, we realized that this model tended to underfit the data. To reduce the bias, our next attempt was with the logistic regression. The difference in score was only marginal.

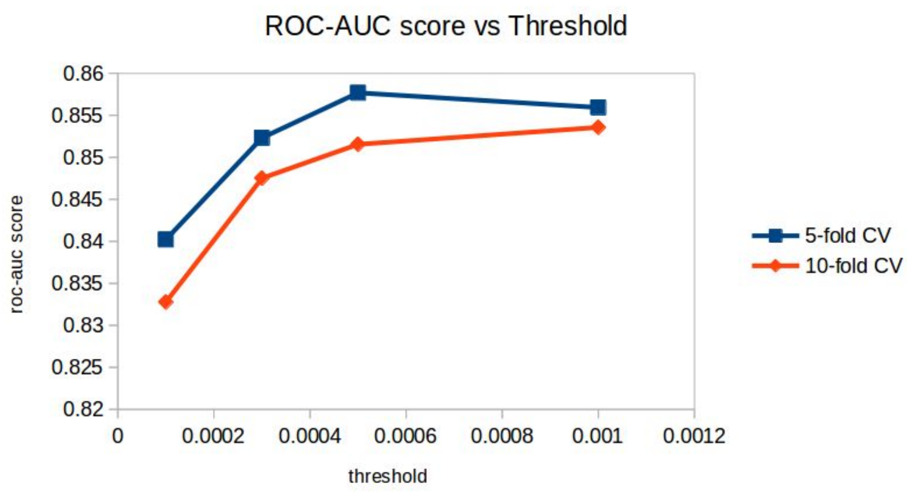
Since logistic regression has more capacity than naive bayes, it should be possible to add more features and improve the overall performance of the classifier. We have used transform function, to transform features in both test and train data to determine



important features. This mayn’t really help for the betterment of classification because it may lead to the either increase or decrease in the roc-auc value depending upon the features eliminated. The final roc-auc score using regularized logistic regression using 10-fold cross validation is shown above figure.

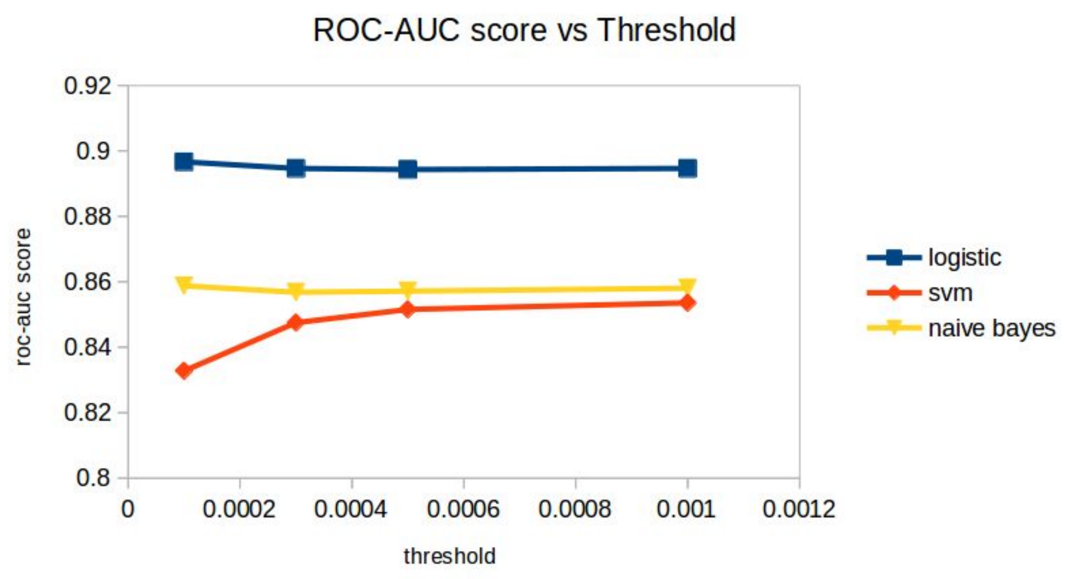
Support Vector Machines:

We implemented SVM with both linear and gaussian kernel. We choose C (Penalty parameter of the error term) as 1.0. In case of gaussian kernel, gamma is considered as auto which means 1/n_features will be used and independent term in kernel function is set as 0.0. Since SVM with gaussian kernel gives bad result as compared to linear SVM in every case for this dataset, the results of gaussian SVM are only included in this report.



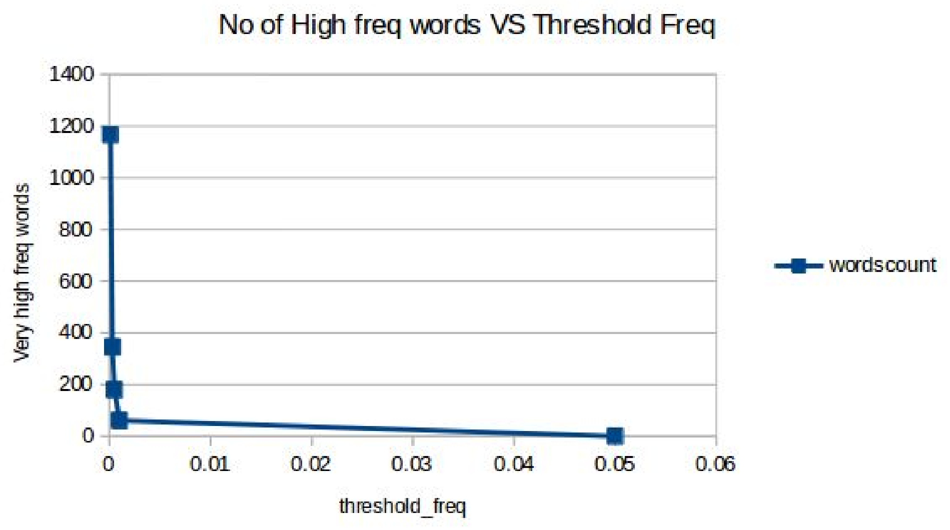
The above graph describes variation of roc-auc score as threshold on fraction of frequency of number of words trimmed changes, on 5-fold and 10-fold cross validation using SVM.

Graph showing performance of various classifiers using 10-fold cross validation:



The above graph clearly shows regularized logistic regression performs better than svm and naive bayes classifiers for our dataset. This pushed us in choosing regularized logistic regression as our final classifier for producing output files.

This graph shows how number of words to trim varies as threshold on fraction of frequency of occurrence of words changes:



Description of Files Generated

These files are generated in the same folder as main.py

- ➢ **preprocessed_train_title.p** contains preprocessed title (after tokenization, trimming stopwords, and stemming or lemmatization as choosed) of each training sample in a serialized format
- ➢ **preprocessed_train_body.p** contains preprocessed body (after tokenization, trimming stopwords, and stemming or lemmatization as choosed) of each training sample in a serialized format
- ➢ **preprocessed_test_title.p** contains preprocessed title (after tokenization, trimming stopwords, and stemming or lemmatization as choosed) of each test sample in a serialized format
- ➢ **preprocessed_test_body.p** contains preprocessed body (after tokenization, trimming stopwords, and stemming or lemmatization as choosed) of each test sample in a serialized format
- ➢ **preprocessed_train_url.p** contains preprocessed url (after stripping digits, stop words and stripping words like: 'http', 'https', 'www', 'com', 'net', 'org', 'm', 'html', 'htm', and stemming or lemmatization as choosed) of each train sample in a serialized format
- ➢ **preprocessed_test_url.p** contains preprocessed url (after stripping digits, stop words and stripping words like: 'http', 'https', 'www', 'com', 'net', 'org', 'm', 'html', 'htm', and stemming or lemmatization as choosed) of each test sample in a serialized format
- ➢ **high_frequency_words.p** contains highly frequent words in both test and train data , which are of not much importance in classification and are trimmed from data
- ➢ **modified_train_data_with_removed_high_frequncy_words.p** contains preprocessed train data after removing highly frequent words
- ➢ **modified_test_data_with_removed_high_frequncy_words.p** contains preprocessed test data after removing highly frequent words
- ➢ **prediction_train_data.csv** contains output predicted values for train data
- ➢ **prediction_test_data.csv** contains output predicted values for test data

Note: In addition to files generated as described above, status of how the code proceeds in getting to the predicted values is displayed on terminal.

Conclusions

The web page classification problem is vital to many web-mining applications and we presented a method to effectively solve the Evergreen vs Ephemeral Challenge. We have determined that the user classification for this dataset has been heavily biased towards food and recipe related websites, and biased against websites related to news articles, technology and sports. It is possible that this particular group of users believes that recipe websites are more likely to be of interest in the long run. An unexpected trend we noticed was that, in none of our algorithms did cleaning up the text (by removing stop words, spelling errors, stemming, removing duplicate words) showed large improvement in the prediction accuracy. In fact in some cases the prediction accuracy actually became slightly worse. This could be because latent factors in the original text, such as the stop words and different forms of a word may have influenced a user's opinion of whether that website can be considered evergreen or not. In future, we plan to extend our model for sentiment analysis of web pages and emails as well as study ways to model noise better.

Future Scope

There is clearly some headroom for improving the performance on the Kaggle / StumbleUpon dataset. We have implemented a number of different classification algorithms, but our results have not improved past 87% regardless of how we changed the parameters for each algorithm. Therefore we propose some obvious extensions to this project which could be implemented at a later time.

- ➢ We have been relying on the data which has been provided to us by the HTML parser, but we believe that we might attain better classification accuracy if we had used the text from the raw HTML files for each website.
- ➢ Since a lot of numeric features had missing or invalid values, we might have implemented other software to compute those fields for each website and incorporated them into our analysis.
- ➢ Instead of relying on classification algorithms being implemented in isolation, it is possible that computing different linear combinations of the outputs of all the algorithms might have given us better accuracy.
- ➢ Using ensemble learning methods should also improve the performance of this system.

This is clearly worth exploring further. In general, information retrieval literature suggests many ideas for improving term scoring such as document length normalization, using different weights for title, URL and body hits.

Individual Contribution

Pre-midstage:

Everyone in the team has equally contributed in studying and researching various algorithms to preprocess data and implementing stemming, lemmatization and url cleaning

Post-midstage:

➢ B Srinivas Naik (140050064): revised mid-stage's source code and completed source code for feature extraction, and classification using regularized logistic regression and SVM, also contributed in documentation and testing
➢ Goutham Arukonda (140050065): source code for Multinomial Naive Bayesian classification and contributed in documentation and testing
➢ Abhinav Rondi (140050054): source code for svm, contributed in documentation and testing
➢ Gurjot Singh Walia (140020121): source code for regularized logistic regression, contributed in documentation and testing

Resources

1. http://cs229.stanford.edu/proj2013/Chen-ClassifyingEphemeralVsEvergreenContentOnTheWeb.pdf
2. http://cs229.stanford.edu/proj2014/Elaine%20Zhou,%20Lingtong%20Sun,%20Evergreen%20or%20Ephemeral%20-%20Predicting%20Webpage%20Longevity%20Through%20Relevancy%20Features.pdf
3. http://www.cs.toronto.edu/~frank/csc2501/Readings/R2_Porter/Porter-1980.pdf
4. http://www.nltk.org/api/nltk.stem.html
5. http://www.nltk.org/_modules/nltk/tokenize.html
6. http://www.eecs.wsu.edu/~holder/courses/cse6363/spr04/slides/ROC.pdf
7. http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
8. http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html
9. http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
10. http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html