Evergreen Classification: Exploring New Features

Shailesh Bavadekar
shail@stanford.edu

Introduction

The advent of internet and social networking has made it very easy to create and distribute content. Discovering the most informative and engaging documents has become a needle in the haystack problem for the consumers. We can divide user information need 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. Often these articles may refer to specific news event at the time of publication, but they may provide a broader perspective that remains of interest well after the event.
    - Literature, arts and entertainment articles. These can be truly timeless.

A robust evergreen content classifier can alleviate this content discovery problem. This report explores evergreen classification using the data set provided by Kaggle StumbleUpon evergreen competition.

Data

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.

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

Naive Bayes Classifiers

The dataset contained both pre-filtered text (without HTML markup) as well as the raw HTML content of each page. These were converted into bag-of-words document models. This process involves many algorithmic and parametric choices such as choice of stemming algorithms, stop words, unigram and bigram terms etc. I tried a few variants with successively better naive bayes classification performance.

NB-1 Classifier

This classifier was built following features:

Evergreen	Ephemeral
egg, cream, food, use, chees, into, sugar, thi, my, add, butter, 4, minut, chocol, cook, make, your, until, bake, or, cup, 2, recip, you	he, hi, sport, fashion, news, video, said, imag, s, game, ha, year, new, technolog, world, who, she, their, team, her, app, show, design, model, flashvar

Table 1: Words most indicative of evergreen and ephemercal classes

- Title, URL and body text fields processed using porter stemmping, converted to unigram and bigram BoW models.
- Unigram and bigram IDF scores were computed as $log_{10}\frac{N}{DF_t}$. Terms with low IDF scores (threshold 0.2) were filtered.
- Laplace smoothing was used to handle unseen terms.

Table 1 shows top 25 stemmed words that are most indicative of evergreen and ephemeral classes. Figure 1 shows the learning curves for this classifier.

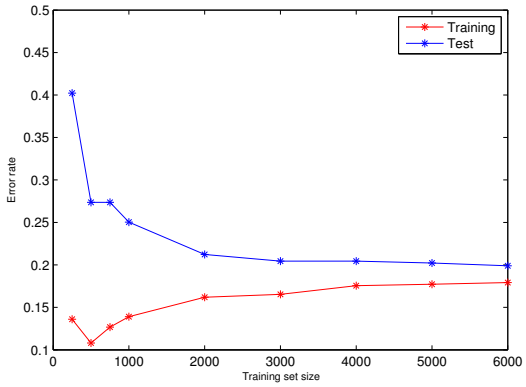

Figure 1: NB-1 Learning Curve

NB-2 Classifier

The NB-1 classifier used only textual features. The StumbleUpon data set also contains quantitative features such as compression ratio, spelling error ratio, image ratio etc. In order to use these features in naive bayes, I bucketized these features and calculated the bucket likelihoods under naive bayes assumption. However the resulting classifier showed negligible improvement in accuracy.

This is in agreement with the learning curve in Figure 1 which suggests a high bias setting. To further confirm this, I also developed an SVM classifier with just the quantitative features and it had a relatively poor performance with approximately 62 percent accuracy. This shows that the raw quantitative features provided in the StumbleUpon dataset have insufficient variance to predict the classes.

Finally, the distribution of tokens in the Table 1 is also very instructive. Many of the tokens (e.g. new, news, world, year etc) that are strongly correlated with the ephemeral class are commonly found in news reports. On the other hand, the evergreen category appears to be dominated by terms related to food and recipes. Approximately 71% of the training documents labelled as evergreen contain one of the terms 'food', 'recipe' or 'cook', while only 16% of ephemeral documents contain these terms. This clearly seems to be an artifact of the sampling process that created the dataset and deserves further investigation.

Logistic Regression Classifiers

Logistic regression models typically work better than Naive Bayes in high bias scenarios. In this section we will explore the feature engineering and performance of regularized logistic regression classifiers.

TF-IDF Features

- This classifier employed the most commonly used TF-IDF functions. Specifically, term frequency was squashed using log(tf) so that longer

documents do not get an unfair influence.

- Unigram and bigram IDF scores were computed as $log_{10} \frac{N}{DF_t}$.
- The lexicon contains 1.5 million distinct unigrams and bigrams (after stemming) drawn from a training corpus containing 6 million tokens.

LR-1 Classifier

Due to large number of features and moderately large number of training documents, I used the liblinear logistic regression with L2 regularization. As the figure 2 shows, this implementation largely fixes the high bias limitation of the NB-1 classifier.

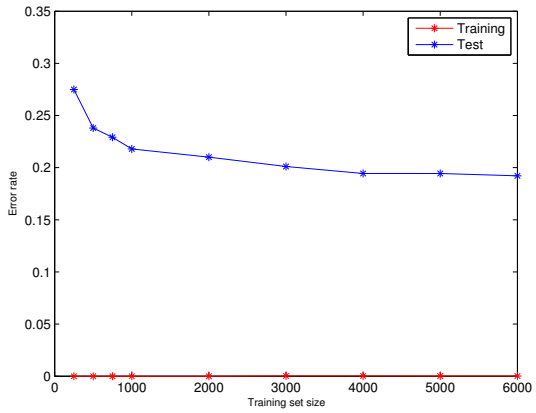

Figure 2: LR-1 Learning Curve

LR-2 Classifier: Visual Page Appearance Features

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. One way to do this is using features that represent the visual appearance of the page. We hypothesize that it is possible for human raters to predict the usefulness of a web page based only on visual appearance of the page without delving into the textual content.

For example, text legibility, placement of images and videos, background color, choice of fonts and colors, etc can have a big impact on the overall quality and utility of a webpage. Furthermore, these features should be independent of the textual features in the bag-of-words models used in NB-1 and LR-1 classifiers. To explore this hypothesis I developed LR-2, a logistic regression classifier that uses only the visual features.

The feature extraction pipeline contains following components.

- The Selenium WebDriver project provides a way to automate browser interaction. This can be used to automate crawling and browser screenshot generation.
- The X.org project provides a tool called Xvfb, a virtual framebuffer server that implements the X window protocol. Xvfb emulates a framebuffer without any physical display hardware. It can also create framebuffers of arbitrary dimensions and depth. Most computer displays do not have enough vertical pixels to render a full webpage. Using 1000x8000x24 framebuffer I was able to create full page screenshots for most training and test web pages.
- These screenshot images were then scaled to a much more manageable size of 330x1959 to reduce the amount of computation.
- The final stage converts the images to greyscale and generates Histogram of Oriented Gradients (HoG) features. HoG features have been successfully used for object recognition tasks.

Figure 3 shows the information represented by the HoG feature. The shape of images and text, the layout of the page can be discerned from the image.
The LR-2 classifier used just the HoG feature vector to classify the documents. The Figure 4 shows the learning curve for this classifier. The accuracy is predictably low, however it provides an intuition about the capacity of this approach to detect evergreen documents.

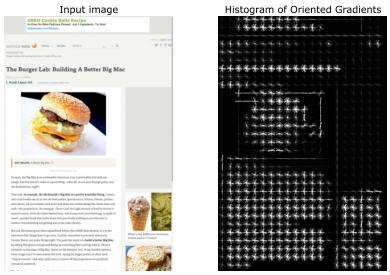

Figure 3: Visualizing HoG features

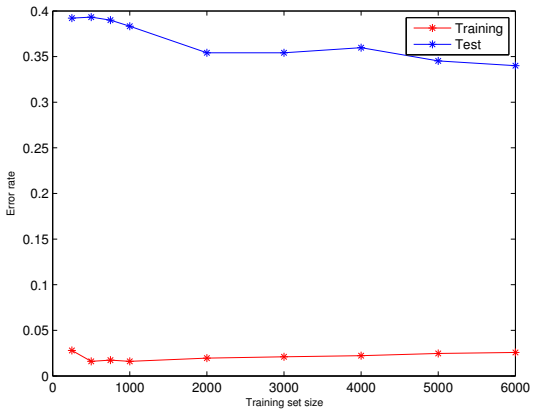

Figure 4: LR-2 Learning Curve

LR-3 Classifier: Text + HoG

The last model in this exploration is a logistic regression model using a combination of textual and HoG features. The figure 5 shows the learning curve for LR-3. It is possible that with more training data LR-3 could continue converging further than LR-2 but the results are inconclusive.

Conclusions

- The accuracy of all classifiers converged to around 80%. Adding more features and better sampled training data should improve the performance.

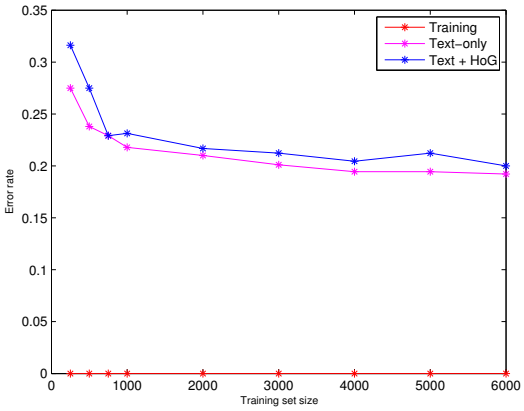

Figure 5: LR-3 Learning Curve

- The HoG features performed better than the quantitative features provided with the Kaggle/StumbleUpon dataset. Using combination of TF-IDF and screenshot HoG features seems promising but more training data will be required to translate that into improved performance.

Future Work

There is clearly some headroom for improving the performance on the Kaggle / StumbleUpon dataset. Following are some ideas that are likely to improve the results.

- I developed a new TF-IDF scoring library for this exercise. Using the standard TfIdfVectorizer class in python sklearn package yielded slightly better results. 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.

- Recent work on vector space representation of words can be applied to this problem. The BoW models can be mapped into a higher dimensional

space which encapsulates meaning / synonyms of the words.

- Both the feature generation and training algorithms have many hyperparameters. For example, the Histogram of oriented gradients feature was generated using downscaled images. This transformation loses information. Using systematic cross validation can yield a better fit for these hyperparameters.
- Using ensemble learning methods should also improve the performance of this sytem.

Even though this dataset has limitations, the ideas described in this paper translate well to a broader class of document classification problems. Classifying documents based on visual appearance has many interesting applications. For example, web design is often conducted in an ad hoc manner in the industry due to lack of resources. The methods described in this report can be easily used to quantify utility and quality of user interfaces.

References

[1] StumbleUpon Evergreen Classification Challenge. `https://www.kaggle.com/c/stumbleupon/`.

[2] Histograms of Oriented Gradients for Human Detection, 2005.