Supervised Online URL Classification with N-Grams

Nickolus Clayton  
California State University, Sacramento  
claytonnick1@gmail.com

Abstract

Under Construction.

Keywords: Web Classification, Web mining, Text classification, Machine Learning

1. Introduction

Some websites regularly release small pieces of information in a way that is not convenient for consumers to keep up to date with. This is an issue that can be easily solved by systems like RSS feeds and email notifications that deliver isolated new content to users as it becomes available. Not all such websites implement a delivery system like this. Thus, there exists a niche for easily configurable applications that can regularly collect these small pieces of content and deliver it to users. Additionally, sometimes the content is time-sensitive and is removed after span of days. Consumer controlled crawler applications could help people to keep abreast of information from a large set of sites and help to preserve old temporary content. In order to create simple software that learns to fill this niche without requiring users to learn a language like regular expressions, we first need an algorithm that can quickly classify various URLs. It would need to sort them into links to ignore, links to follow, and links to collect, based on a small amount of labelled data provided by the user.

2. Related Work

Many previous works in the realm of URL classification take advantage of the structures present in URLs to extract useful features. UPA is one such algorithm [1]. It functions similarly to a decision tree whose features are the strings that exist between various punctuation and delimiters in URLs. It seems wiser though, to base a URL classification algorithm on more generic text-based classifiers to better account for information in the text of the URL and to enable the inclusion of related text outside of the URL. Many more recent researchers have been thinking along similar lines. The n-Gram Language model is particularly popular due to its simplicity and flexibility. This method dices text into overlapping substrings of length n and counts their relative frequencies. This enables creation of vectors of n-gram frequencies which can be classified via traditional approaches like Bayesian logic and support vector machines. One brute-force variant of this is referred to as all-grams. Instead of analyzing text substrings of a specific size n, all-grams employs substrings of every possible length. Though it is a strong technique for identifying patterns in data, the memory required to store the substring table is prohibitive. The Log-Likelihood Model is based on Bayesian logic applied to a vector of n-grams. It’s makes a compromise between a naïve Bayes classifier and correct Bayesian logic by judging the likelihood of a classification for each character given the previous n-1 characters. Abdallah and Iglesia’ experiments showed that the accuracy of this method is comparable with all-grams but requires only twice the space of a typical n-gram data structure [2].

3. Method

It is possible to use URLs alone to classify the majority of URL sets that consumers would need since existing related software like Comica use regular expressions to
classify URLs [3]. I wrote scripts to collect two appropriate sets of URLs, convert them into 3-grams, and store them in standard data tables. They are unbalanced and have 376 and 951 dimensions respectively. These tables are used to test the following algorithms: naïve Bayes, support vector machine, Manhattan-distance clustering, and log-likelihood odds. In these tests I have an expectation of 100% accuracy given sufficient data as the datasets are devoid of noise and contain human-recognizable patterns. This is not cause for concern though since these tests are to establish both their maximum accuracy and speed of generalization. In order to test speed of generalization, the data sets are randomly divided into n segments of equal size so that training is performed on each individual segment in turn and testing is performed with the remaining n-1 segments. I will make modifications to the algorithms as necessary to better suit the data.

4. Experiments

Clustering Easy Dataset (to be converted to table form)
Poll = 1, Partiton Size = 5, 83.3519530881484 % accurate
Poll = 1, Partiton Size = 10, 90.366279250927 % accurate
Poll = 1, Partiton Size = 15, 93.4299650677345 % accurate
Poll = 1, Partiton Size = 20, 95.1425780701015 % accurate
Poll = 1, Partiton Size = 25, 96.2649886691162 % accurate
Poll = 1, Partiton Size = 30, 96.5589353612167 % accurate
Poll = 1, Partiton Size = 35, 96.8169372811207 % accurate
Poll = 1, Partiton Size = 40, 97.0711594511973 % accurate
Poll = 1, Partiton Size = 45, 97.0515911282546 % accurate
Poll = 1, Partiton Size = 50, 97.3172757475083 % accurate
Poll = 1, Partiton Size = 55, 97.2295392815046 % accurate
Poll = 1, Partiton Size = 60, 97.1548162644053 % accurate

Clustering Hard Dataset (to be converted to table form)
Poll = 1, Partiton Size = 5, 63.0229419703104 % accurate
Poll = 1, Partiton Size = 10, 76.2094698919197 % accurate
Poll = 1, Partiton Size = 15, 80.3411776795807 % accurate
Poll = 1, Partiton Size = 20, 79.8006644182722 % accurate
Poll = 1, Partiton Size = 25, 80.719375177866 % accurate
Poll = 1, Partiton Size = 30, 81.4386056191467 % accurate
Poll = 1, Partiton Size = 35, 83.2528180354267 % accurate
Poll = 1, Partiton Size = 40, 83.3858543417367 % accurate
Poll = 1, Partiton Size = 45, 84.354272337105 % accurate
Poll = 1, Partiton Size = 50, 84.1861598440546 % accurate
Poll = 1, Partiton Size = 55, 84.8508481185416 % accurate
Poll = 1, Partiton Size = 60, 84.9388379204893 % accurate

Large dataset accuracy test results:

naïve bayes (easy, 75% training set): 72.7%
native bayes (difficult, 75% training set): 0.9%

support vector machine (easy, 75% training set): 100%
support vector machine (difficult, 75% training set): 100%\}


Clustering and LLO have been implemented. Clustering needs more work as I still have ideas for improving it. LLO needs to be tested formally.

naïve bayes (easy, 75% training set): 72.7%
native bayes (difficult, 75% training set): 0.9%

support vector machine (easy, 75% training set): 100%
support vector machine (difficult, 75% training set): 100%\}
Both naïve Bayes and naïve clustering fail these initial tests due to the way they value each 3-gram equally, and do not account for their interdependence.

**Speed of Generalization:**

support vector machine (easy, small set): 97.2% accurate
support vector machine (difficult, 3.6% training set): 80.2% accurate
4.1. Difficulties

Temporary

The Scaling function in R cannot accept columns that have no variance. Both clustering and LLO needed small modifications to account for the unbalanced nature of the training sets. I have decided to include a second URL representing the page on which the other URL was found in a separate space from the first URL. This enables the theoretical crawler software to explore websites in a more directional manner. Most of the misclassified data points from the LLO model arise from a cruder implementation of this where the URLs are merely concatenated with a separator.

5. Conclusion

10. References

