10-701 Machine Learning, Fall 2014
Interim Report

BSBot: A Learning System to Detect Fraudulent Abstracts in High Energy Physics

Dustin Axman, daxman
Baihu Qian, bqian
Connor Walsh, connorw

1 Introduction

Content-free grammar has enabled several "almost-real" publication generators for several areas such as mathematics, computer science, and high-energy physics. Content-free grammar is a set of rules that are independent from the words (or phrases) they apply to. By defining a dictionary containing field-related keywords, and a set of rules similar to languages used in research publications, it can generate any number of publications. Famous examples of such generator is SCIgen, an automatic CS paper generator; snarXiv, a random high-energy theory paper generator similar to arXiv; the Real Theorem Generator for mathematical theorems, and Philosophy of the Day, a philosophy sentence generator.

Both SCIgen and snarXiv have proved that human are not good at distinguishing computer-generated papers from real papers. One of the SCIgen-generated paper was accepted by WMSCI 2005, an influential multi-discipline conference in computer science, and snarXiv reports that classification between real paper titles and machine-generated titles by human is merely 59%, slightly higher than random.

Therefore, we aim to build a classification mechanism to distinguish machine-generated contents to real, human-produced contents. There are very few publications on this topic. A SCIgen paper detection method was described in this paper by using inter-textual distance (based on bag-of-words) and knn classification method. The author claims it should "hardly" misclassify, but no performance metric is given.

We focus on paper title and abstract because snarXiv can only generate these two parts. Thus far, we have crawled data from arXiv.org, providing positive data points; acquired the source for the snarXiv abstract generator to generate negative data points; developed a framework for informed feature selection; and have begun evaluating our feature transform methods.

2 Data Acquisition

As no previously compiled dataset existed for the problem of interest, it was required that we collect and preprocess the relevant data by our own means. Despite the inconvenience, this allowed for flexibility in formatting and data selection obviating the need to filter irrelevant and superfluous data fields often found in inherited datasets. Consequently, we were able to produce conformed formatting across both positive and negative data samples. Here, we are primarily interested in the textual content of abstracts, both real and false, and their accompanying titles.

Acquiring negative data points involved obtaining, compiling, and executing the snarXiv generator code. The code is freely available courtesy of Whats-his-name (his-website) and includes a grammar file, `snarxiv.gram`, and a perl script, `compile-grammar.pl`, used to compile the final OCaml source, `snarxiv.ml`. Executing this program produces a single pseudo abstract with a title, list of authors, and subject header. Implementing our own extraction python script, `generateSnarXivData.py`, we were able to extract the titles and abstract bodies of any specified number of pseudo abstracts.

For the collection of our positive data points, it was necessary to mine honest abstracts from the theoretical high energy physics database of ArXiv. This required the development of a rudimentary web-crawler, `arXivCrawnler.java`, to chronologically gather a user defined number of data samples from the most recent to the oldest archive entries. The mean collection time ($n = 1000$) for a single sample is 0.427 seconds with a standard deviation of 0.363 seconds.

The final results from both collection programs are stored in `.txt` files for each sample. Within each sample file, there is one word per line with the title and abstract bodies delimited by double forward slashes.

3 Approach

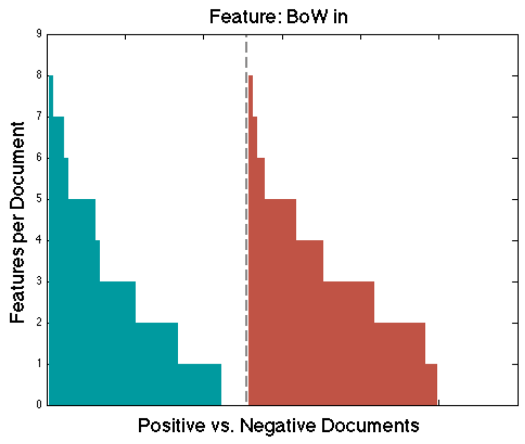
3.1 Feature Selection

Naturally, when presented with a dataset comprised wholly of text strings, the prevailing response is to employ a *Bag-of-words* approach in feature selection in which a dictionary of chosen words or phrases are used to discriminate between multiple document classes. While this feature selection method certainly has merit, the information it conveys is limited due to the loss of structure imposed during feature transformation. For this reason, we incorporate additional features which preserve some syntactic structure in order to exploit our prior knowledge of context-free grammar.

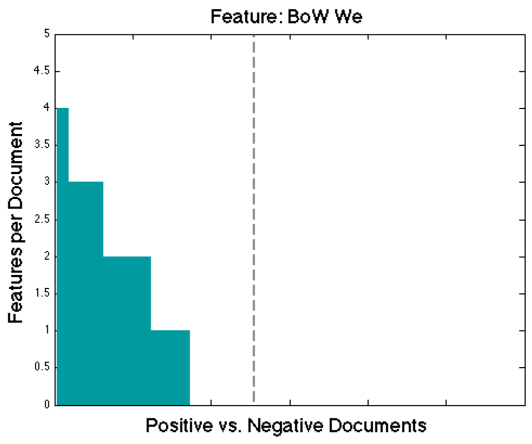
We also implement a series of lexical parsers. For example, we use NLTK to do part of speech tagging in each sentence for each document sample. This allows us to track the counts of each tag as a separate feature. We generate a sentence tree for each sentence. This allows us to use features that include the depth and breadth of the sentence to judge complexity overall for each document. One interesting extra feature drawn from this is the ambiguity of noun reference. When a sentence is referentially ambiguous, two trees are generated. This allows us to judge the average incidences of referential ambiguity in a document and use this as a feature. This feature, combined with sentence complexity, should be effective at discriminating between the two classes. The Corpus used for lexical parsing is the generic Brown Corpus though we plan to experiment with several others.

While carefully premeditating the types of features to include in the final learning system is crucial for an accurate classifier, often it is unclear which features will yield optimal performance. For a binary linear discriminant learner, the accuracy of performance manifests itself as the degree of separability between classes of high dimensional features. Thus, it is absolutely critical to evaluate the candidate features with respect to their separability between classes.

To this end, we have developed a method to filter the best features in terms of the information they convey for class discrimination. The method we present draw on previous work and is designed to evaluate generic feature types. The general idea is to choose features which display a large amount of class imbalance, whether this be in the frequency count per corpus or the number of corpora which contain a given feature. For example, fig 3.1(a,b) shows a feature (the frequency of the word "in") with poor separability and a feature (the frequency of the word "We") with strong separability.



3.1 (a)



3.1 (b)

To select for the best features, we formulate the problem as follows,

$$f_h^* = \underset{f}{argmax}\ D_h(X_f^+, X_f^-) \quad (1)$$

$$f_I^* = \underset{f}{argmax}\ D_I(X_f^+, X_f^-) \quad (2)$$

where f_h^* is the optimal feature which maximizes a hamming distance, D_h, between positive and negative training examples, X^+ and X^-, transformed by feature f. Similarly, f_I^* is the optimal feature which maximizes an intersection distance, D_I.

For D_h we define the distance as follows,

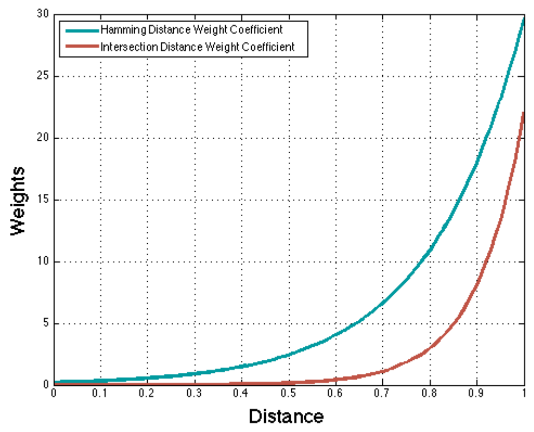
$$D_h = \sum |X^+_{\{x>0\}} - X^-_{\{x>0\}}| \tag{3}$$

which is the absolute valued distance between binary vectors, $X^+_{\{x>0\}}$ and $X^-_{\{x>0\}}$, which indicate whether each document has a non-zero feature count or not. For D_I we use the tanimoto distance. Both distance measures return values in $[0, 1]$.

In addition to using the distance measures to choose the most appropriate features, we also use the values in a weighting scheme employed by our feature transform. We define the feature transform as follows,

$$f_h = sign(max(X^+, X-)) \cdot exp(a_0 D_h)/a_1 \tag{4}$$
$$f_I = sign(max(X^+, X-)) \cdot exp(b_0 D_I)/b_1 \tag{5}$$

where the constants (a_0, a_1, b_0, b_1) are chosen. This transform provides weights which increase exponentially as the distances grow and ensure that the weights have the proper sign corresponding to the class imbalance of the feature. Fig 3.1(c) shows an example of the feature transform function.



3.1(c)

4 Preliminary Result

Having completed the development and implementation of the feature selection method described above, our initial results show that *Bag-of-Words* features are providing the best separation between classes. This, however, does not imply that we have abandoned hope on syntactic and lexical type features. Instead, we are interested in continuing to test a more diverse set of these types of features. Interestingly though, at this point we have not found any BoW features which are exclusively indicative of a given class. Rather, we have been able to identify a large amount of well partitioned features whose scope within the class they are partitioned to is relatively small. This means that we will most likely be working with a high feature dimension.

5 Conclusion and Future work

So far we have gathered features and found the most determinant ones for classifying between generated High Energy Particle Physics Abstracts and real, written High Energy Particle Physics Abstracts. We did this through a series of methods we created, as described above, to weigh the features based on the hamming distance between their frequency graphs. With these weighted features we will be able to find a metric for effectively judging the performance of several classification schemes on our data. We are well prepared to continue using this data to do an in depth evaluation of several classification methods beginning with SVMs. With the feature extraction code we have already implemented it is easy to add more features to our list and judge them very efficiently.

In the future we plan to add many more types of features. We also plan on finding or creating other separation methods to judge against. These metrics include other feature sets and classifiers such as Logistic Regression and KNN grouping. We also plan to find out how well humans do in classifying these Abstracts. Our initial information on this says that humans do a very poor job of classifying the abstracts. If we can achieve a higher correct classification percentage than humans then that will be a good first step. We plan on implementing sentence complexity by analyzing the syntax and tagged trees and finding their depth and width. We need to establish a grammar file to compute these trees effectively and this will take some time but should be a very attainable future goal. We also plan on implementing other counts of tagged words. We have looked at several kernels as well. Many of these string kernels work on k-mers and the hamming distance between these k-mers and the words that they are derived from. We will experiment with several of these kernels and use cross validation over our training set to find the one most effective in establishing separation.