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. We have crawled and formatted arXiv papers and grouped into months, and generated snarXiv papers. Currently we use . The classification accuracy is .

We plan to use for the next step, and explore performance variations across time.

This report

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 us 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 will 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 margin 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 [**?**]. DISCRIBE HISTOGRAM METRICS.

3.2 Models

4 Preliminary Result

5 Conclusion and Future work

References