**Documentation for automatic classification of stories**

**CPEG 666: Independent study**

**University of Delaware**

Jan Sher Khan

**Introduction:**

In this project a sample data from [Github](https://github.com/vz-risk/VCDB/issues) is taken, Which contains news stories about cyber-attacks/crimes. Each story has been categorized with a tag by the humans. The name of the tags are as follow: Breach, PHIDBR2018, Misuse, Autocode-test, Physical, Malware, Defacement, DOS, Hacking, pinboardimport, HOF-2014, Social, Mining, POS, Error, Outage, Update. There are around 4 thousand stories and more are added to the website on regular basis.

**Objective**:

To analyze the text of each story and how they are related to their respective tags.

**Procedure:**

The procedure is broken down into small steps given below:

**The first step** of the procedure involves the collection of data/stories. The stories should be collected in a way that, each story’s URL, date of addition, its tag, Tag provider(Human) and name of the file stored in the system. The name of each story stored in the system is also by the date of addition of the respective story. Information about all the stories is stored in the form of a CSV format.

**The second step** involvesthe sorting of data in the form of test set and training set. 10 different combinations of data having the distribution of 20% and 80% of test and training respectively.

The combinations can also be named as folds for ease of understanding.

**The third step** involves the conversion of downloaded HTML code in a plain text. For this purpose, the text-mode browser is used by the name of “[Links](http://links.twibright.com/about.php)”. It converts all the HTML code in a plain readable text.

**The fourth step** involves the grouping of training set in their respective tags.

**The fifth step** involves the text processing of the grouped sets. Before the proper processing, the unwanted characters are removed from the stories like: \*,●,\_ etc. Removing the stop words and word-stemming can also be introduced but are not applied in this project. After filtering, the text is converted into 1 gram till 10 grams. The 10 high frequency grams of each N-grams are taken so in this way we get 10\*10 grams or 100 items from each tag-category of the training set

**The sixth step** involves making of 10\*10 grams of an individual story from the test set and comparing each story with the, 10\*10 grams, tags of the training set. In this comparison the number of common items/characters with each tag are counted/noted. In this way every individual story is compared with every tag from the training set and the comparison result is noted in the form of CSV output.

**The seventh step** involves applying of the sixth step to the all remaining 9 folds. In this way we get 10 csv files.

**The eighth step** is working on the resulting CSV files. The story which has a ground truth tag has greater number of match with similar tag but in the training set, compared to remaining tags, is considered as success. The number of success are counted for each story and percentage is taken called success percentage.

**The Ninth step** requires the applying of eighth step on the remaining CSV’s from each fold. An overall average of success percentage of all the folds is calculated, which in our case, resulted in 38% of success match.

**Tools used in project**

**Python:**

Python programming language (v2.7) is used for this project.

**Beautiful Soup:**

“[Beautiful Soup](https://www.crummy.com/software/BeautifulSoup/)” is a Python package used to parse and retrieve the stories from the [Github](https://github.com/vz-risk/VCDB/issues) website. Installation via command line, “$ pip install beautifulsoup4”

**Links:**

[Links](http://links.twibright.com/) is a web browser running in both graphics and text mode with comparatively less execution time. It is a command line browser.To convert the HTML code into text. [Links](http://links.twibright.com/) is an optimum tool in this situation because it neglects the images or some other scripts and the user gets the text form only. [*Installation guide*](http://links.twibright.com/user_en.html#subch-compilation_and_installation)*.*

**Python’s NLTK:**

The Natural language processing toolkit ([NLTK](http://www.nltk.org/)) is a Python’s package, which comes handy while processing the natural language. It also has the libraries for classification, tokenization, stemming, tagging and parsing. Installation via command line, “$ pip install NLTK”

**Python’s Pandas:**

[Pandas](http://pandas.pydata.org/) is a Pythons package used to process the .CSV files efficiently. It is a powerful data analysis toolkit. Installation via command line, “$ pip install pandas”.

**Python OS:**

The [OS](https://docs.python.org/2/library/os.html) module allows users to use the operating system dependent functionalities. The Links browser is used in the command line interface of an operating system. The [OS](https://docs.python.org/2/library/os.html) module can do this job while in python. Installation via command line “$ pip install os”.