Will Diedrick

CS 467 – Machine Learning

Professor Nina Marhamati

Final Project Report

**Wikipedia Vandalism Detection**

**Application Development:**

Wikipedia is an open source project that relies on the collaboration of many users to build Wiki-pages and edit Wiki meta-data. Since it is an open source project, anyone can edit this kind of information in a simplistic manner. However, not all editors have the best intentions when using the Wikipedia API. It is crucial to remove any information that might be considered vandalism from Wikipedia pages. Bayesian Inference allows for a preventative solution that targets the words used by potential editors. Bayesian inference is a method in which Bayes' Theorem is used to update the probability for a hypothesis as more evidence or information becomes available (Gotze).

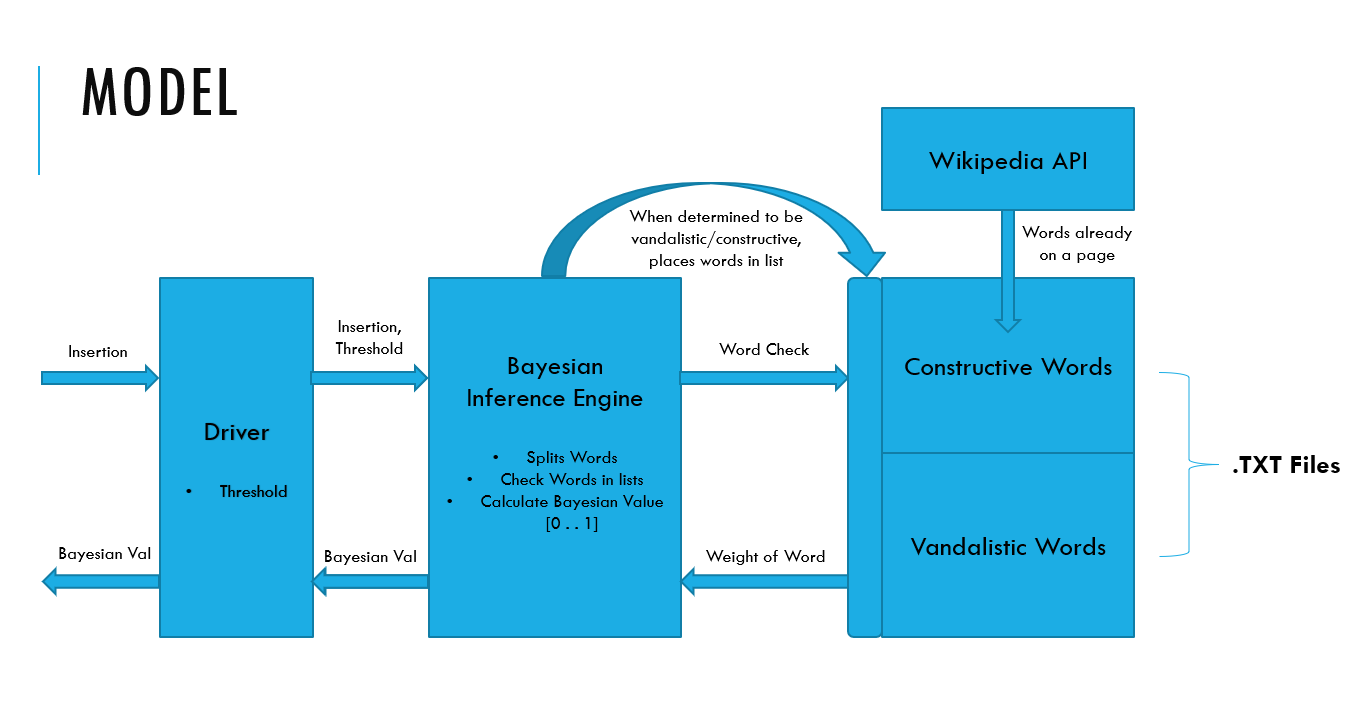
**Problem Definition:**

*“How can we make sure that the information being placed onto a specific Wikipedia page is constructive and not vandalism?”* With this project, the target page will be the Darrell Kruger Library Wikipedia page.

**Project Goals:**

* The goal of this project is to design a program that can successfully define whether an insertion to a specific Wikipedia page is vandalism or constructive.
* It should learn from any type of edit that is made by adding words from the insertion to the appropriate dataset (blacklist/whitelist).
* The system should employ fuzzy logic; Any result is based on the scale of [0 . . 1].
* A Bayesian Formula should be used to calculate the final return value.
* Any improvements to the design should be conceptualized and recorded.

**Model Formulation:**

 The model design was based around the instructions of a Bayesian Inference Engine and how it interacts with the database that holds the vandalism and constructive words. The Bayesian Inference Engine holds the portion of the program that does the computation for the Bayesian value. It completes this by splitting an insertion into single words, comparing the words to the word lists to obtain a weight for the word, and by using Bayes’ Theorem to calculate the value based off the word weights. The driver class is used to provide a threshold value for the Bayesian Inference Engine; The threshold will be used to determine whether the Bayesian value should be considered constructive or vandalism. Based on the conclusion of the engine, it will place the insertion into either the vandal words or constructive words database. This process will simulate “learning” in the model.

**Implementation:**

The implementation of this project is based upon the model that was designed. The implementation process was completed using the tools found within the Wikipedia API and Java libraries. The target for protection is the Wikipedia page for the Darrell Krueger Library from Winona State. The driver, Bayesian Inference Engine, and word storage are all separated to organize the integral parts of the program. The driver accesses the Bayesian Inference Engine and passes in the insertion and threshold (0.5). Using these values, the engine will systematically check every word with the established lists of constructive and vandalistic words implemented through .TXT files. The constructive list will hold all of the words from the Wikipedia page being protected. The Wikipedia page can be downloaded in a .XML format from the Wikipedia API. Two counters are kept for every match made in the separate lists. The program will calculate a ratio [0 . . 1] of words which are considered constructive to the total times the words show up in the lists. If the word does not appear in either list, it is inherently considered to be vandalistic and assigned the value of (0.3). Certain words deemed unbiased will also hold no weight in the program. Examples of such words are “is”, ”and”, and “the”. The ratio for each word is then added together and divided by the total amount of words in the insertion to achieve a Bayesian value. If the Bayesian value is above the threshold, the inserted words will be placed onto the constructive text file. If it is below the threshold, the words are placed on the vandalism text file. The Bayesian value is then returned to the driver and outputted. The program is implemented through the use of arrays and word counts which place target words onto .TXT files. The more words that are placed onto the .TXT files, the stronger the expected inference of the program is.

**Computational Study:**

With the knowledge of how the model was created and implemented, a computational study of the program can be initiated. This program is a machine learning project, so changes in the results of the program will occur. This change occurs when words from insertions are placed onto the constructive and vandalistic .TXT files, changing the counters and weights behind each word. It is expected that words relating to the page, like “Library”, “Book”, and “Resource” will hold high weights. Words unrelated to the page, like “Pizza”, “Rainbow”, and “Geology” will hold low weights. This system allows for easy mistakes to be made, especially when it comes to false positives. Certain words that could be considered vandalistic could be given higher weights if they are associated with words that are constructive. To avoid this, one should make sure that the vandalism list is prepared with all words that could possibly be used in a vandalism attack.

A way to dramatically improve the performance of this system is to implement a Neural Network with the Bayesian Inference Engine. The Neural Network can take in the Bayesian value as well as other values to compute a better interpretation of the insertion. A spelling value could be implemented, giving words with incorrect spelling a lower weight. A size of insertion value was also considered, insertions with a less amount of words would be given a lower weight. This is based on the premise that most vandalism would be short and not as well thought through.

This program also has no method for sorting out punctuation and spelling errors. All insertions must be correctly spelled words with a single space separating them and no punctuation. The storage design of the program currently uses .TXT files with high redundancy when it comes to the number of words stored. Instead of the .TXT files, an SQL database with words and their related weights could be implemented to store a sufficiently large amount of space. A filter could also run through the program to randomly adjust some of the weights so that the detection service does not get stuck in a certain pattern.

**Conclusion:**

The method of Bayesian Inference for vandalism protection worked well for the test cases it was put through. However, it is not hard to trick the system into adjusting the weights of certain words to become more vandalistic. The best way to improve the performance of this program is to construct a solid set of vandalistic and constructive words before it is used. ClueBot NG is a real-world example of this kind of program that is currently being used on Wikipedia servers. ClueBot NG makes use of the Bayesian Inference model as well as the use of a Neural Network to create a probability that an insertion is vandalism (Breneman). ClueBot NG is a working system that proves the utility of a project like this.

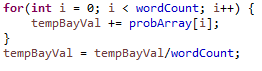
**Software Development:**

This program required the design and implementation of a program that can detect vandalism from insertions onto the Darrel Krueger Library Wikipedia page. This section will go more in depth to the software side of the program and how to operate it. The program was written in Java based around the concepts outlined in ClueBot NG’s code. This program is functionally different from ClueBot NG because it deals with the weights of specific words where ClueBot NG deals with the weights of insertions in general. To operate this program functionally, the word storage units must be prepared. The constructive words list need to contain all the words on the target Wikipedia page.

**Algorithm Description:**

The algorithm used within this program can be described as an average of the collective probabilities that each word is constructive. These probabilities are found by dividing how many times it occurs in the constructive list by how many times it occurs in the constructive list and the vandalistic list. Division shown:

After the probabilities for each array is stored in the probability array, this array is then added together and divided by the amount of words in the insertion.

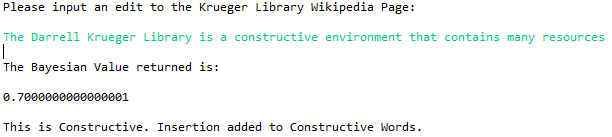


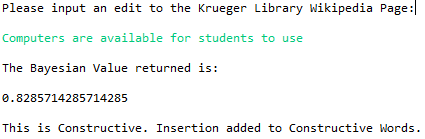
This algorithm will return a value between [0 . . 1] which is the Bayesian value. It interprets how likely the insertion is constructive, 0 being completely vandalistic and 1 being completely constructive. A threshold is necessary to determine the exact point where something is constructive. In this program our threshold is 0.5.

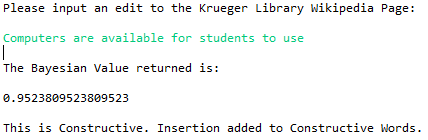
**User’s Manual:**

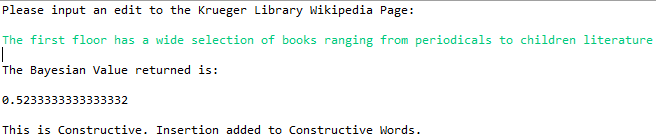
Upon initiating the program, the user will be prompted for an insertion they wish to put on the target Wikipedia page. This insertion must be spelled correctly, and any punctuation will result in incorrect values. Capitalization does not matter as the words are all converted to lower case before they are compared with the lists. A threshold value must be decided before the program is initiated. It can be changed from within the driver class. Keep in mind that a higher threshold means less words will be considered constructive. Upon entering the insertion into the program, it will produce a Bayesian value and whether or not the insertion is considered to be constructive or vandalism. Repeatedly entering the same insertion into the program will result in the Bayesian value changing; This is the program “learning”.

**Example problems:**

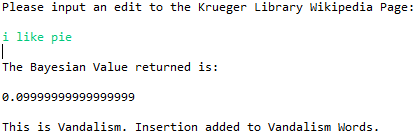
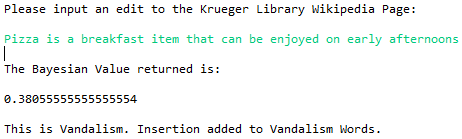
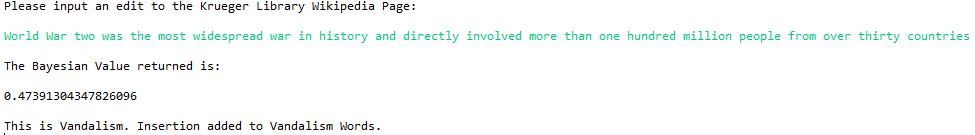
 **Constructive Insertions:**

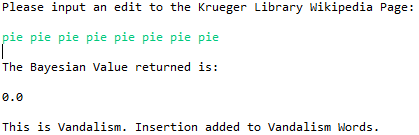




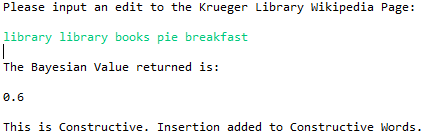


**Vandalism Insertions:**

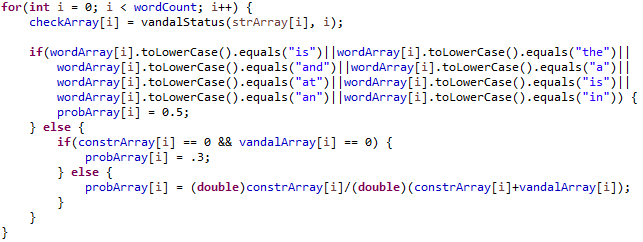




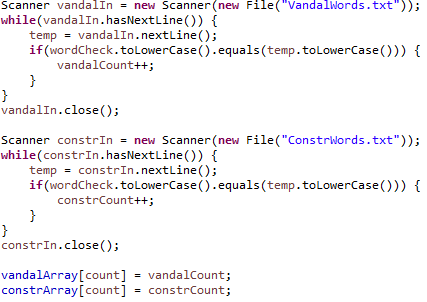
**LoopHole:**



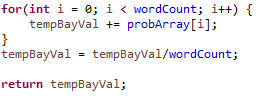
**Computer Code Descriptions:**

This code splits a phrase into an array of Strings, each String is a word.

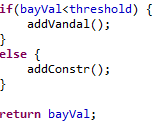
This code will take every word and give it a probability based on how many times it appears in the constructive list and vandalistic list. If it is any of the words listed in the first if statement it will be ignored. If it shows up in neither list, it is assigned a probability of 0.3.

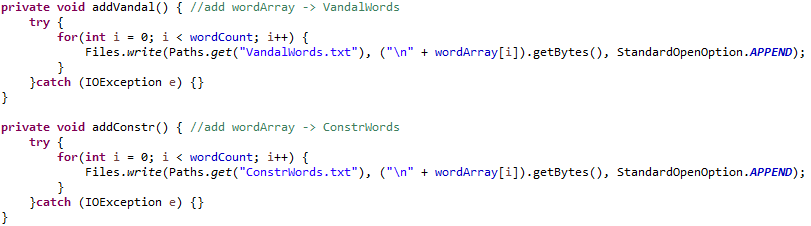


This code takes a word and counts how many times it occurs in both the constructive and vandalism lists.



This code will compute the Bayesian value based on the probabilities of all the words.



This code will determine whether the insertion is constructive or vandalistic depending on whether the Bayesian value is higher than the threshold.

This code will write to a specific file depending on whether the Bayesian values is constructive or vandalistic.

**Citations:**

Adler, Thomas. “Combining natural language, metadata, and reputation features.” *Wikipedia vandalism detection*, Springer-Verlag, dl.acm.org/citation.cfm?id=1964776.

Breneman, C., Carter, J., Tim1357. “ClueBot NG.” *Wikipedia*, Wikimedia Foundation, 28 Nov. 2017, en.wikipedia.org/wiki/Wikipedia:Bots/Requests\_for\_approval/ClueBot\_NG.

Gotze, Paul. “Advanced Vandalism Detection on Wikipedia.” 3 Sept. 2014. http://www.uni-weimar.de/medien/webis/teaching/theses/goetze\_2014.pdf

“Krueger Library.” *Wikipedia*, Wikimedia Foundation, 23 July 2017, en.wikipedia.org/wiki/Krueger\_Library.