Vandalism Detection in Wikipedia

Dillon Caryl
Computer Science
University of Texas at Austin

Vivek Khetan
School of Information
University of Texas at Austin

Edward Babbie
School of Information
University of Texas at Austin

Juliette Seive
School of Information
University of Texas at Austin

ABSTRACT
In this paper, we investigate machine learning approaches to the Wikidata Vandalism Detection Competition.

Keywords
Wikipedia, vandalism, detection, information retrieval, deep learning, machine learning

1. INTRODUCTION
This project explores the Wikidata Vandalism Detection competition [1] and the plan for implementation of this exploration. After having read extensively about deep learning and how it is different from other forms of machine learning - sometimes better and sometimes not, each member of our group became curious about *how* to implement what was being discussed in the literature. The English version of Wikipedia alone receives around 3 million edits per month, and approximately 7% of edits to wikipedia are vandalism [3][4]. Needless to say, discovering and reverting these changes automatically would save significant time and effort on the behalf of Wikipedia editors. The goal of the project is to create a simple deep learning model to detect vandalism as well as to explore the literature that surrounds the current vandalism bots Wikipedia uses and how, based on the articles read in class, deep learning can increase detection and reduce vandalism.

2. RESEARCH QUESTIONS
What is the current state of limitations and vandalism detection in Wikipedia? How good is deep learning at recognizing vandalism in text edits? Can other deep learning methods offer an alternative, or optimization to the current WikiBot model? These questions clamor for the practical application of deep learning in the realm of natural language processing, which has been predominantly run by machine learning classifiers. With strides being made in many disciplines through the implementation of deep learning, it begs the question of if it is possible for the same impact to be made in the area of vandalism detection.

Wikipedia houses millions of articles that are accessed daily from around the world, and deep learning has yet to make its presence known in the field of vandalism detection. With it, there is the potential to avoid the storage of damaging and erroneous claims throughout the online encyclopedia.

We believe that these questions are important for the reasons stated above as well seeing whether or not deep learning can play an active role in typical NLP tasks, and if a simple deep learning model can rival a simple machine learning model. Additionally, a literature review of how Wikipedia currently handles vandalism will aid in creating a better understanding of the current landscape of monitoring and detection solutions. It will also offer us the ability to better craft and formulate a superior implementation based on our research into the literature on deep learning and the current state of Wikipedia's vandalism detection policies and practices.

2.1 Prior Work
Initially, Wikipedia vandalism was handled by manually monitoring changes using tools such as Vandal Fighter [11]. As Wikipedia grew, this approach alone became unwieldy, so simple rule-based bots such as ClueBot and AVB (Anti-Vandal Bot) became more prominent to allow automatic reversion of obvious vandalism, such as nonsense strings of characters [8]. Wikipedia currently uses ClueBot NG, a bot based on machine learning, to automatically revert vandalism [5]. Unfortunately, there appears to be a large divide between the practical side of volunteer vandalism bot creation and formal academic research. We were unable to find research discussing these bots other than as a historical or tangential observation [8].

Our first known research using machine learning on a Wikipedia vandalism corpus was Potthast et. al in [7] who created a simple feature set and trained it on a manually collected corpus of ~1000 edits. Velasco [9] extended Potthast's feature set with categories such as vulgarisms, biased words, and "good" words. Around this time, it became clear that a more extensive, public vandalism corpus would be required to promote standardized research. Potthast [3] created the first large-scale, public Wikipedia vandalism corpus we are aware of using Amazon's Mechanical Turk. Later, Heindorf et. al [10] created a series of more extensive, formalized corpuses for the related Wikidata project, which is a structured data source used in Wikipedia and other sites.

Our most modern source, and the one provided as reference for the Wikidata Vandalism Detection competition, is Heindorf et. al in 2016 [6]. Their model uses a random forest to learn 47 features and beat the state-of-the-art Wikidata Abuse Filter.

All of these studies are discussed in greater detail in the Section 4, the Literature Review. We will also find more sources to bridge the gap between early methods and Heindorf et al. in 2016.

3. METHODOLOGY
Our project is split into two parts: implementation and literature review. Before introducing our methods for these two parts, understanding the data is of importance. We originally planned on using the training data provided by the competition, but given the difficulty of pre-processing, we decided to move forward with a corpus built specifically for detecting wikipedia vandalism. We split this corpus in order to create a testing dataset and to perform cross validation.

For the implementation portion we built two simple machine learning classification models (a Logistic Regression Classifier and Stochastic Gradient Descent Classifier) and are working on a simple deep learning model. The purpose of the machine learning

model is to create a benchmark for our deep learning model. The first model, albeit simple, will allow us to judge whether or not the latter model can compete with what is considered the industry standard today. This method is beneficial not only because it affords us the ability to compare the models, but also because it allows us as a team to receive a hands-on understanding of implementing the models, the differences between the two, and how these differences may ultimately result in a performance distinction for the vandalism detection problem.

The literature review provides us the opportunity to experience an important balancing act in conjunction with our implementation model. Because our time to build out the models is limited, we hope that a literature review will help us understand 1) how Wikipedia currently implements their vandalism detection bots and 2) which aspects of deep learning are beyond our scope of implementation but may ultimately increase accuracy and spark interest and excitement for further investigation and research. While there is value in fleshing out models, there is equal importance in understanding the current state of things and offering informed suggestions, which is what we hope to contribute to the deep learning space with our two-fold project.

4. LITERATURE REVIEW

This literature review will discuss a practical history of anti-vandalism automation on Wikipedia, the creation of formalized vandalism training corpuses for ML, and research on ML for vandalism detection.

4.1 Practical History of Anti-Vandalism efforts on Wikipedia

The earliest tools for detecting and reverting vandalism in Wikipedia were simple diff-based implementations which allowed trusted editors to monitor changelists in real-time. One such example, Vandal Fighter, also allowed administrators to sort and filter the edits by users, whitelists and blacklists, regular expressions, and other methods [11]. In another example, WikiScanner simply identified anonymous editors with their IP addresses to identify recurrent cases of lobbying [7]. Vandal Fighter and most subsequent tools and bots operate on a special Internet Relay Chat (IRC) channel that effectively streams data about changes as they occur in real time.

Vandal Fighter and a variety of similar tools made at least three things abundantly clear:

1. There were far too many instances of vandalism to manually keep up with.
2. The majority of vandalistic edits are extremely obvious (random strings of characters, excessive use of profanity, caps lock, and so on). A significant number could be reliably detected with simple rules [12].
3. Metadata regarding users was highly effective at predicting vandalism. Blacklists and whitelists were amongst the most used tools in anti-vandalism toolkits [11].

To that end, around 2006, anti-vandalism and anti-spam bots began making their first reversions on Wikipedia. Amongst the most prolific were AVBOT (2008-2010), which made around 600,000 edits using a scoring system of regular expressions; ClueBot (2007-2010), which used a scoring system based on heuristic rules to make around 1.5 million edits; and SaleBot (2007-present) which used user profiling to remove ~700,000 instances of vandalism [12].

Using ClueBot as a case study, we can see the types of rules implemented by these early bots. Smets et. al [8] describe this fabulously:

> "ClueBot, written by Carter (2007), uses a number of simple heuristics to detect a subset of the types of vandalism mentioned above. First, it detects page replaces and page blanks relying on an auto-summary feature of MedaWiki software. Next, it categorizes mass delete, mass addition and small changes based on absolute difference in length. For the last three types, vandalism is determined by using a manually crafted static score list with regular expressions specifying the obscenities and defining some grammar rules which are hard to maintain and easy to by-pass. Negative scores are given to words or syntactical constructions that seem impossible in good articles, while wiki links and wiki transcludes are considered as positive. The difference between the current and the last revision is calculated using a standard diff algorithm. Thereafter, the inserted and deleted sentences are analysed using the score list and if this value exceeds a certain threshold vandalism is signaled. ClueBot further relies on the user whitelist for trusted users and increases its precision by only reverting edits done by anonymous or new users."

Wikipedia's current state-of-the-art is ClueBot NG (2010-present) with 1.6 million edits so far [12]. ClueBot NG uses an artificial neural network to assign a probability that a given edit is an act of vandalism. ClueBot uses supervised learning on a dataset labeled by Wikipedia volunteers, in conjunction with Bayesian classifiers, to calculate a probability that a given edit is an act of vandalism. Finally, humans set the thresholds to classify an edit as an automatic revert, a case for further human review, or a valid edit. These thresholds have occasionally been tweaked to achieve an appropriate balance between detection and false positive rates [5].

4.2 Vandalism Training Corpus Creation

4.2.1 "Crowdsourcing a Wikipedia Vandalism Corpus." - Potthast 2010

Wikipedia has grown and thrived from the collective knowledge and input from people everywhere. In order to continue this progress while keeping in the spirit of crowdsourcing, a large corpus of human-annotated edits was developed in [3] to help work towards automatic vandalism detection. Amazon's Mechanical Turk was utilized to find workers that would help label the 32,452 available edits as vandalism or legitimate. Groups of 3 would annotate an edit, and if greater than 2/3 agreement was not reached, 3 more annotators would provide their feedback. This would continually happen until a consensus was reached. The final result left 2,391 of the edits marked as vandalism.

4.2.2 "Towards Vandalism Detection in Knowledge Bases: Corpus Construction and Analysis" - Heindorf et. al 2016

As the usage of search engines has become increasingly prevalent, the necessity of a reliable knowledge base offering support has never been so apparent. Unfortunately, these public bases receive

millions of revisions each month, and there currently is no feasible way to automate an integrity check. To fill this void, Wikidata was chosen as the knowledge base to provide the first standardized vandalism corpus for the backend for Wikipedia, and a corpus analysis was performed in [10] to determine which pages are being vandalized and who is doing the performing the edits.

Wikidata currently stores about 50 million subject-predicate-object triples and 17 million items, where items are grouped by subject and each item gets its own page. Each edit of an item is assigned a unique ID and stored within Wikidata's revision history. For this corpus, only manual edits are considered, which leaves around 24 million available and only 103,205 being labeled as vandalism.

After accumulating this data, it became much easier to grasp who the vandalizers are and what they are vandalizing. A little less than 60 % of the vandalism is performed on textual data like labels, aliases and descriptions. What's more interesting though is the difference between percentage of edits that are vandalized vs percentage of total edits for each item's category. For example, 20% of vandalized edits are to items in the Culture category, but only 12% of the edits are made in the Culture category. Conversely, 31% of the total edits are made in the Places category, but only 8% of the vandalized edits are to the Places category. Culture receives less attention, but more vandalism. A last, perhaps unsurprising discovery is that 86% of the vandalism on Wikidata is from unregistered users, and only 41% of these edits originate from a previously used IP address. Anonymous users tend to make 1.7 vandalistic edits while registered users tend to make 6.4 vandalistic edits on average.

Vandalism has been an ever-growing thorn in the side of Wikidata (and consequently, Wikipedia), and if left unchecked it could result in massive amounts of misinformation to reach the public in all corners of the world. Fortunately, with a corpus that spans the entire history of Wikidata and lessons learned from extrapolating the data, an effective, automatic labeling strategy has never looked more possible.

4.3 ML Approaches

4.3.1 "Automatic Vandalism Detection in Wikipedia" - Potthast et. al 2008

In one of the first ML-based approaches to vandalism detection in Wikipedia, Potthast et. al [7] formally introduce the "Vandalism Detection Task." They define a vandalism corpus *E* that is a set of edits, where each "edit" is comprised of two consecutive copies of a document *d* on wikipedia. The task is then simply a classification task in which each edit in *E* can be classified as either a case of vandalism or a regular edit.

They also implement a number of features that form the basis for future models. These features were manually engineered by looking at 301 vandalism examples to identify common themes. In particular, they identify three types of vandalism: lobbyists, who promote bias in articles; spammers, who introduce advertisements or attempt to sell products; and vandals, who "deliberately destroy the work of others." Then, they specify features that can be divided into character-level, word-level, edit-level, and user-level. These features can be seen in the following table:

Table 1: Potthast et. al's feature list from [7]

Table 3. Features which quantify the characteristics of vandalism in Wikipedia

Feature f	Description
char distribution	deviation of the edit's character distribution from the expectation
char sequence	longest consecutive sequence of the same character in an edit
compressibility	compression rate of an edit's text
upper case ratio	ratio of upper case letters to all letters of an edit's text
term frequency	average relative frequency of an edit's words in the new revision
longest word	length of the longest word
pronoun frequency	number of pronouns relative to the number of an edit's words (only first-person and second-person pronouns are considered)
pronoun impact	percentage by which an edit's pronouns increase the number of pronouns in the new revision
vulgarism frequency	number of vulgar words relative to the number of an edit's words
vulgarism impact	percentage by which an edit's vulgar words increase the number of vulgar words in the new revision
size ratio	the size of the new version compared to the size of the old one
replacement similarity	similarity of deleted text to the text inserted in exchange
context relation	similarity of the new version to Wikipedia articles found for keywords extracted from the inserted text
anonymity	whether an edit was submitted anonymously, or not
comment length	the character length of the comment supplied with an edit
edits per user	number of previously submitted edits from the same editor or IP

Most of these features are fairly self-explanatory. Unusual character combinations tend to represent gibberish. First person pronouns and vulgarism tend to indicate unprofessional edits. Replacement similarity and context can potentially identify irrelevant additions, and, as we have seen, user metadata can often identify users whose posts should all be reverted. One of the more interesting features is compressibility, where most legitimate English edits should fall into a certain range of compressibility, and deviations from this indicate gibberish or repetition. The researchers then train these features using a basic logistic regression algorithm.

Unfortunately, no real vandalism corpus exist at the time of this paper, and while Wikipedia's volunteers had reverted a lot of vandalism, there were no explicit labels for vandalism reversions. The researchers instead create their own corpus manually by looking at 940 edits of Wikipedia's most commonly vandalized pages and identifying 301 cases of vandalism.

They compare their results with all features against AVBOT and ClueBot. They vastly outperform in recall with results ranging from .76 for replacement (vs .53 and .29) to .89 for deletion (vs .61 and .49). Their precision outperforms at .86 vs. AVBOT's .74. Their baseline instance of ClueBot has been configured to set its precision threshold at 1, but it has much lower recall. The authors argue that their performance profile is preferable as it allows the machine to catch most vandalism in a two-pass approach of machine-human monitoring. While the results are decidedly impressive, we question the validity of feature engineering, training, and testing all based on the same small dataset of 1,000 samples. It is very likely that the model suffers from overfitting. It may also be that the distribution of edits in this corpus, chosen from the most commonly vandalized pages, will differ from edits in the wider Wikipedia ecosystem. For example, it is likely that the incidence of vandalism is much higher in this corpus (at ~30%) and those instances of vandalism may be of specific varieties (such as lobbying on the pages of controversial political figures).

4.3.2 "Wikipedia Vandalism Detection Through Machine Learning: Feature Review and New Proposals" - Mola-Velasco 2010

Mola-Velasco's primary contribution in [9] is to directly add to the features just proposed in [7] and test them on the 15,000 sample crowdsourced corpus in [3]. To that end, he adds character-level features to detect the ratio of numeric and special characters, which can be indicative of spam or nonsense. He also adds the absolute change in size (which can indicate large deletions) and makes various modifications to existing features in [7], such as testing different compression algorithms to find that most useful algorithm for the compressibility feature.

He also elaborates on a number of word-level categories for use in features at that level. These include vulgarisms, first and second person pronouns, biased words (such as subjective superlatives), non-vulgar sex words (which are highly indicative of vandalism in some contexts while totally normal in other contexts, such as biology), and "good" words which are rarely used by vandals, especially syntax like "<ref>."

He uses a decision tree as a baseline and implements these features in both a random forest and SVM, although he discards the latter due to its computational requirements. He gets the best results from a random forest of 1000 trees with recall 0.861, precision 0.568, f-measure 0.684, and "Area Under ROC Curve" of 0.963. This is a moderate improvement over the decision tree baseline of 0.773, 0.543, 0.638, and 0.93, respectively. Unfortunately, there are no direct comparisons to the bots shown in [7] on this dataset, but the algorithm won first place in the PAN'10 vandalism detection competition. Mola-Velasco concludes that precision is not high enough to let the bot run automatically in the real-world, due to the number of false positives, but that it shows promise.

4.3.3 "Vandalism Detection in Wikidata" - Heindorft et. al 2016

In [6], Heindorft et. al propose an advanced ML approach to vandalism detection and test it on the Wikidata corpus created in [10]. Wikidata was launched in 2012 and presents a structured database of facts, such as the birth dates, educational history, and families of famous people, often found in tables on the right side of Wikipedia articles. This structured relationship makes the problem similar to the Wikipedia vandalism detection problem, but slightly different in that Wikidata statements are uniformly structured, clear in their relationships to one another, and accessible through a standardized API.

Their model uses a random forest trained on 47 features, which include many of the features mentioned in [7] and [9]. Some of the most notable additions at each level are:

- Character-level: "random keystrokes, character repetitions, wrong or excessive capitalization, missing spaces, and the use of special characters (e.g., for emoticons)." [6]
- Word-level: ratios of upper and lower case words, ratios of "bad words", and ratios of words from other languages
- Sentence-level: primarily focused on the "sentences" given in edit comments, including the length and relevance of the comment

Additionally, Wikidata's structured nature allows the researchers to more clearly analyze relationships between different pieces of data, which they call "statement-level" features. Since most of the properties defined in Wikidata are links to other Wikidata pages, the graph between them should follow certain patterns. For example, President Obama's page has a property linking to Joe Biden as his vice president , and vice-versa. These properties tend to have specific frequencies of links versus literals. Pages also have "popularities." For example, the "Democratic Party" page is linked to very often, as all Democrats have it as a property. So these relationships and the complex network built up between the pages can be used in a series of features.

The authors also claim that they are the first to consider multiple-instance learning, which treats edits not as individual entities, but rather as part of a session in which a user makes multiple edits in one sitting. As predicted, edits within a single session have even higher correlations than simply considering edits by the same user, so this contributes significantly to prediction performance. The authors claim that combining multiple-instance learning and bagging more than doubles their model's performance over their default random forest [6].

Their results are impressive and show a major improvement over their baselines, ORES and FILTER, which are the abuse filters currently used in Wikidata. Their measurements are area under curves (AUC) of both the receiver operating characteristic (ROC) and the precision-recall curve (PR). Their most optimized approach maintains an ROC of ~.99, compared to ~.90 and ~.98 for FILTER and ORES. More relatably, their PR measure is ~.49 compared to ~.22 and .~23. This shows the magnitude of their improvement in real-world detection performance in both precision and recall.

5. DATA PRE-PROCESSING

Before being able to begin any model building, the data offered by the Wikipedia Vandalism Detection Competition had to be processed. In this section we review how we attempted to clean the Wiki data, our shift to another dataset, and the data munging steps we took to create a useable data frame ready for modeling. The majority of our work thus far has been spent on the data pre-processing of the original Wiki Data before migrating to the Wiki Vandalism Corpus last week.

5.1 Wiki Data

The Wikidata Vandalism Corpus 2016 consists of 21 manual revision datafiles, 1 metadata file and 1 truth file. We downloaded the first manual revision file (smallest in size, 18MB) and uncompressed it to get our xml file.

Understanding and converting this data to a pandas DataFrame or a csv file was really tough job. The data was very dirty and had a lot of missing as well as wrong entries and the deep layer of xml also added to the complexity of the task. We were able to write a python script to convert our xml file to a pandas DataFrame for doing data analysis. The dataset we converted was not able to fulfill our purpose (to train a Machine Learning algorithm) as there was no True example for vandalism in it.

5.2 Wiki Vandalism Corpus

After struggling with the initial competition-provided data set, we decided to work with a corpus created specifically for Wikipedia vandalism detection. This data set is presented in a ready-to-consume format. The PAN Wikipedia vandalism corpus 2011, created by a group of researchers dissatisfied by the existing data, features "29949 [English language] edits on 24351 Wikipedia articles, among which 2813 vandalism edits [were] identified" [13]

supplying us with an acceptable amount of data to fulfill our goals of building simple machine and deep learning models.

The corpus was downloaded and each CSV file was transformed into a pandas dataframe for further manipulation. From here the columns containing text, such as "articletitle" and "comments", were cleaned, which included removing special characters, punctuation, and lowercasing all of the words. Additionally some of the data frames were merged together to create a single data frame with all the desired features we wanted to work with during the model creation process. We are in the process of adding another column that changes the labels for vandalism ("vandalism versus "regular") to a binary 1 (vandalism) or 0 (not vandalism). We will re-run the model accordingly.

6. MACHINE LEARNING MODELS

In order to compare our deep learning model, we ran two machine learning models - a Logistic Regression and SGD Classifier - to establish a baseline. Both of these models are simple in the interest of time, so that the majority of the remaining time can be spent on developing and understanding a deep learning model.

6.1 Logistic Regression Classifier

Our first baseline model is a Logistic Regression Classifier. We split the 80% of data as training set and 20% as our testing set. Data cleaning and wrangling took a lot of effort and time. We converted the yes/no vandalizing label to True/False label, deleted many rows with lots of NAN values.

The next important part was to identify the important features and to create new required features for vandalism detection. We use Labelbinarizer technique for various categorical features and created new binary features corresponding to each category. We also created a duration feature, i.e., the time between the start of the edit to the time when the editor submitted the edit. Then, we vectorised comments and articletitle columns and created 1000 and 500 new feature respectively.

We created two different dataset, one dataset (Dataset1) without the 1500 features created from comment and articletitle columns another dataset (Dataset2) with these new 1500 columns. We dropped old comment and article id column from both the dataset. We trained Logistic Regression model to get True/False as the prediction output and it was giving an accuracy of 99 percent.

This was due to class imbalance problem in our datasets. We had very few vandalism examples in comparison to not vandalism. We adjusted the class weights depending upon the number for samples in each class and this resulted in an accuracy of 71.9 percent for Dataset1 and 72.3 for Dataset2. Next, we will also make a SGD model to compare with Logistic Regression model and create a better baseline for our deep learning model.

6.2 Stochastic Gradient Descent Classifier

Our second baseline model is an SGD Classifier. Our data was split into a training and testing set where both sets have labeled targets. Before running the model, our main text features ("articletitle" and "comments") were vectorized into word embeddings. We extracted a total of 16568 features for this model with 70,000 examples.

During model creation, we assigned weights to each class in order to attempt to combat overfitting, if for example, an article title is common. The model was fit using the data features described above and the 70000 "gold annotations" (whether an edit was considered vandalism or not vandalism) for the examples previously mentioned.

For testing, the remaining 70,000 examples were vectorized to create data features and fit to the SGD model. The annotation results were predicted using these data features. The model performed well when predicting the "non vandalism" labels but not very well when predicting the "vandalism" labels. One possibility for these results is that in the training data there are many more "non vandalism" examples than the converse. We are also currently working on running with model with "predict probabilities" as a means to get more granular with the model's certainity on its labeling. With our current implementation, on average, the model performs as follows: precision is at 74%, recall at 84% and f1-score is at 78%. We believe this model presents an acceptable, simple baseline to use to compare against a deep learning model.

We believe that one method to improve the model would be by including additional examples from another Wiki Vandalism Detection corpus which are in languages other than English [14]. An increase in examples - in other words more data - would help us increase our model's ability to better predict "vandalism" labels. There are many more features that we could consider and spend time engineering, but since our main goal as a team is to apply and understand how deep learning works in a typical machine learning tasks, we are moving forward accordingly.

7. DEEP LEARNING MODEL

After creating and evaluating our machine learning models, it became necessary to build an accurate deep learning model so as to make the comparison. Again, with time being a major constraint, the plan is for the model to be fairly simple, yet effective in demonstrating the capabilities of deep learning in the realm of vandalism detection.

7.1 Technology Used

In order to achieve our goals, we have opted to use Google's open source software library known as TensorFlow. This library is commonly used for machine intelligence and can be tuned in order to meet our goals. It utilizes data flow graphs that feature nodes and edges. The nodes represent mathematical operations whereas the edges are used to symbolize the data arrays, or tensors, that are communicated between the nodes. In our case however, the nodes will be altered to be an area for data to be fed in and for variables to be read and written to. The nodes execute asynchronously and in parallel as the data arrays flow along directed edges as it progresses through the graph. Once all of your data is fed in and the computations are conducted, you can then use TensorBoard to visualize the graph you have created.

7.2 Methodology

TensorFlow affords the opportunity to provide input data via multiple methods. Python code can provide the data at each step, an input pipeline can read from files at the beginning of each graph, or for small data sets there can be pre-loaded constants or variable. We have opted to read from a file. All our our data, including the edits made by various users from around the world

and the labels of incorrect or correct edits will be fed into TensorFlow when the graph is originally being created before the computations are made. Now, depending on the file format, a certain reader is needed. Our data is preserved and stored in a comma-separated value format (.csv). Thankfully, there is a clearly outlined method for feeding this type of data into our graph. There is a TextLineReader that can output the line of a file delimited by new lines. Once fed through the reader, there is a native operation that can convert CSV records into tensors. One column is mapped to one tensor. The operation takes in one row from the csv file and in return will provide a list of Tensor objects that are connected together. The data will now be in a format in which our various computations can be conducted and performed.

7.3 Text Classification CNN

In order to gain a better understanding of how a deep learning model is implemented, our team drew inspiration from Kim Yoon's paper "Convolutional Neural Networks for Sentence Classification" [15] where a simple CNN model is implemented with slight hyperparameter tuning. We additionally utilized the TensorFlow tutorial documentation [16]. Our own CNN model uses an embedding layer followed by a convolutional, max-pooling, softmax, and readout layer. Given our group's minimal exposure to deep learning and the learning curve it presented for us, we strove to understand the basics and implement a simple model explained in further detail in the following sections.

A first step in building the CNN model was selecting hyperparameters. We set the embedding dimensions to 128, filter sizes to 3,4, and 5 respectively, the number of filters to 128, and a regularization lambda to 0. While most of these parameters are the defaults, we understand the effect that these have on model accuracy especially in cases where the dataset is small (much like ours) or has quirks. Additionally we set our training parameters to the default batch size of 64 and number of training epochs to 200. One deliberate choice that we did make was to apply dropout before the readout layer as a means of minimizing the overfitting we noticed was prone to happen given our small number of vandalism examples.

The training portion of our deep learning experience was relatively straightforward, as our data is stored in CSV files, for reasons mentioned in section 7.2. The data was already pre-processed (stripped of punctuation and special characters, and lowercased) from our Logistic Regression and SGD models. The model was trained on the same features (article title and comments) as the machine learning classifiers in order to remain consistent and allow for a more accurate comparison during the evaluation section of the project. We did run into one problem during the training of our model - the problem of time. Our relatively small dataset took multiple hours to run. While this was not too much of an inconvenience for us, it highlighted the problem that bigger datasets will likely run into, especially when run on smaller processors. This experience brought into direct light the question of how researchers and developers will handle the issue of efficiency, especially as the amount of available data increases.

We proceeded to test our model on the remaining 20% of our dataset. The most challenging aspect of testing the model was loading the metagraph saved from training, as well as conceptually understanding the representation of our feature columns as tensors and how to select the prediction tensor we wanted to evaluate. The text classification CNN produced an accuracy of 67.44%. Though the accuracy is significantly lower than the SGD classifier, it does perform better than chance with only the default hyperparameters.

Taking a deeper dive into the model's label predictions, we noticed that about 84% of the time it incorrectly predicted that an edit was vandalism when it was non-vandalism. Each incorrect vandalism prediction occurred when the comment associated with the edit said, "reverted edit to last revision". However upon further exploration of the data from the Wiki Vandalism Corpus, not all edits associated with these comments were marked as vandalism. This observation presents a possible area of future work, as well as a need for further investigation and understanding of the metadata supplied by the Wiki Vandalism Corpus.

It is also interesting to observe the model overfit in the direction of false positives even though there are significantly more non-vandalism examples than vandalism ones in the corpus. This idiosyncrasy offers another potential area to focus technical future work on. We believe that some hyperparameter tweaking may help with this issue.

While this deep learning text classification CNN performed less well than the machine learning classifiers, it is apparent that with pointed attention to hyperparameters, a larger, more class-balanced dataset, and appropriate weights for each label, there is reason to further explore this space. Our model has little customization but with more time and knowledge we believe that deep learning could offer a successful alternative to current Wikipedia vandalism detection.

8. EVALUATION

We evaluated ourselves based on our performance in the following tasks:

- A cogent analysis of current Wikipedia vandalism and all other damaging edits to wiki data
- How Wikipedia vandalism bots work and if it can be made more efficient by optimizing some parameters
- To develop simple Machine Learning Classifiers (Logistic Regression and SGD) and understand how good these classifiers are in detecting vandalism, and what reasons we believe are cause for their performance (whether good, bad, or average)
- To develop a Deep Learning Model to detect Wikipedia vandalism, and compare it to the benchmark ML Classifiers
- To gain a group wide better understanding of how deep learning can be applied to typically machine learning tasks and areas of opportunities for future research
- Focusing on finding new and interesting questions related to vandalism and its detection

Overall we feel that our team managed to successfully complete the goals we set for ourselves, as well as delving into our research questions. Though we did not fully answer all these questions given our time and experience constraints, our work allowed for many interesting insights. We managed to produce a literature review, which aggregates the Wikipedia vandalism detection work of others and offer a simple technical example of how machine and deep learning can handle the vandalism detection task.

Challenges arose throughout the project. The original dataset proved to be too difficult to work with under out time limit, but as

a result we were able to use the Wiki Corpus that inspired other Wikipedia vandalism research. Class imbalance and the size of our dataset is another area of the project that could and should be improved on for any future work. All of these challenges and discoveries helped our group better understand the real-life constraints of deep learning and machine learning as a whole.

The main knowledge that we did gain from this project, however, is that the task of identifying vandalism is not a trivial task. There are many components of an edit (the time, comment left, article, timezone, username) that may impact whether or not it should be marked as vandalism. A feature engineering process of this sort is highly time consuming. This problem is part of why we believe that deep learning offers a valuable opportunity for the Wikipedia world, as this feature engineering could be forgone. It is important to note that even as new deep learning developers, we were out-of-the-box able to get an accuracy score of 67.44%, only 5% behind our Logistic Regression model. We believe that this highlights the potential of a more custom text classification CNN.

9. DIVISION OF WORK

Juliette and Vivek have prior experience with ML and set up the baseline detection algorithms (including the pre-processing of data) using a Logistic Regression Classifier and Stochastic Gradient Descent Classifier. Dillon performed the literature review to guide our project. Eddie researched TensorFlow to help us prep for the deep learning model building portion of our project and helped Dillon with the literature review. Juliette built a text classification CNN for the deep learning portion of the project.

10. CONCLUSION

With our two-fold approach of a literature review and model application, we hope to have provided a simple but comprehensive overview of the Wikipedia vandalism detection problem. In the comparison between the machine learning and deep learning models, we believe that we have started to show how deep learning could affect the way Wikipedia detects vandalism.

11. REFERENCES

[1] Web Search and Data Mining Cup 2016: Wikipedia Vandalism Detection (2016) [Online, accessed 10-October-2016] http://www.wsdm-cup-2017.org/vandalism-detection.html

[2] B. Thomas Adler, Luca de Alfaro, Santiago M. Mola-Velasco, Paolo Rosso, and Andrew G. West, "Wikipedia Vandalism Detection: Combining Natural Language, Metadata, and Reputation Features", Lecture Notes in Computer Science: Computational Linguistics and Intelligent Text Processing 6609, 277-288. February 2011. http://repository.upenn.edu/cgi/viewcontent.cgi?article=1494&context=cis_papers

[3] Potthast, M.: Crowdsourcing a Wikipedia Vandalism Corpus. In: Proc. of the 33rd Intl. ACM SIGIR Conf. (SIGIR 2010), ACM Press (Jul 2010). http://dl.acm.org/citation.cfm?doid=1835449.1835617

[4] Wikimedia Foundation: Wikistats (2016) [Online; accessed 26-October-2016]. https://stats.wikimedia.org/

[5] ClueBot NG (2016). [Online; accessed 26-October-2016]. https://en.wikipedia.org/wiki/User:ClueBot_NG

[6] Stefan Heindorf, Martin Potthast, Benno Stein, and Gregor Engels. 2016. Vandalism Detection in Wikidata. In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16). ACM, New York, NY, USA, 327-336. DOI: https://doi.org/10.1145/2983323.2983740

[7] Martin Potthast, Benno Stein, and Robert Gerling. 2008. Automatic vandalism detection in Wikipedia. In Proceedings of the IR research, 30th European conference on Advances in information retrieval (ECIR'08), Craig Macdonald, Iadh Ounis, Vassilis Plachouras, Ian Ruthven, and Ryen W. White (Eds.). Springer-Verlag, Berlin, Heidelberg, 663-668.

[8] K. Smets, B. Goethals, and B. Verdonk. Automatic Vandalism Detection in Wikipedia: Towards a Machine Learning Approach. In Proc. of WikiAI at AAAI'08 https://www.aaai.org/Papers/Workshops/2008/WS-08-15/WS08-15-008.pdf

[9] S. M. Mola-Velasco. Wikipedia Vandalism Detection Through Machine Learning: Feature Review and New Proposals: Lab Report for PAN at CLEF 2010. CLEF Notebooks 2010 https://arxiv.org/abs/1210.5560

[10] S. Heindorf, M. Potthast, B. Stein, and G. Engels. Towards Vandalism Detection in Knowledge Bases: Corpus Construction and Analysis. SIGIR 2015 http://www.uni-weimar.de/medien/webis/publications/papers/stein_2015h.pdf

[11] Vandal Fighter. [Online; accessed 26-October-2016]. https://en.wikipedia.org/wiki/User:CryptoDerk/CDVF

[12] Anti-Vandalism Bot Census. [Online; accessed 15-November-2016]. https://en.wikipedia.org/wiki/User:Emijrp/Anti-vandalism_bot_census

[13] PAN-WVC-10. [Online; accessed 10-November-2016]. http://www.uni-weimar.de/en/media/chairs/webis/corpora/corpus-pan-wvc-10/

[14] PAN-WVC-11. [Online; accessed 10-November-2016]. http://www.uni-weimar.de/en/media/chairs/webis/corpora/corpus-pan-wvc-11/

[15] Yoon Kim. Convolutional Neural Networks for Sentence Classification. Empirical Methods on Natural Language Processing 2014. https://arxiv.org/abs/1408.5882

[16] TensorFlow Documentation. [Online, accessed 10-November-2016] https://www.tensorflow.org/versions/r0.12/tutorials/mnist/pros/index.html#build-a-multilayer-convolutional-network

APPENDIX:

Juliette wrote the Text Classification CNN, Evaluation, and Conclusion sections. She also updated and edited the Data Pre-Processing (intro and Section 5.2), Machine Learning Model (intro, and section 6.2), and Methodology sections.

Vivek wrote the wikidata (5.1) and programmed and wrote the logistic regression classifier (6.1).

Dillon found references and wrote the Prior Work section as well as sections 4, 4.1, and 4.3 of the literature review. He also set up and formatted the paper.

Eddie wrote the Deep Learning Model section and investigated libraries and technologies that can assist in the deep learning implementation. He also contributed to the literature review by completing 4.2.1 and 4.2.2.