|  |
| --- |
| **Low-Cost Methods for Increasing Classification Accuracy** |
|  |
| **Katharine Sadowski1, Linda Wang2**  **1 Department of Policy Analysis and Management, Cornell University**  **2 Department of Economics, Cornell University**  [**ks2373@cornell.edu**](mailto:ks2373@cornell.edu)**, lw692@cornell.edu** |
|  |
|  |
|  |
|  |

Abstract

Classification is a popular task in Natural Language Processing that has large implications, especially for tasks such as classifying whether a comment is toxic. State of the art models, such as BERT, have pushed the accuracy of classification models close to 100%; however, most models are still hovering below this benchmark. We are interested in exploring low-cost solutions that can improve accuracy for researchers interested in classification tasks.

Introduction

In a world where online platforms provide anonymity to users, hateful and toxic language is becoming more prevalent. As this behavior takes off, platforms such as Facebook, Twitter, Reddit, to name a few, are increasingly working on moderating this behavior. There are many methods these companies can consider. Reddit, for instance, uses human moderators to assess whether a comment is against the community guidelines. If they deem it is, it is removed. In addition to this, other platforms such as Facebook, have begun using AI such as BERT in order to classify language as against community standards or not (Facebook blog). This allows the platform to use fewer people, thereby saving the company money.

While both of these features have their advantages, they are either very costly in terms of time or still not 100% accurate. We are interested in other low-cost methods for improving classification accuracy of toxic speech. We believe by harnessing both of these methods, human annotation and BERT, we can increase accuracy with little effort.

Our ideas for this paper are influenced by a variety of other projects that have looked at using either hand annotation or other more technical methods for improving classification accuracy. Researchers have worked on supplementing BERT with their own encodings (HurtLex). While others have crafted their own examples related to common heuristics that were often misclassified (Hans). While these types of methods are quite interesting, they require a lot of expertise and time from the individuals developing the datasets to supplement our machine learning methods. Other researchers have attempted to limit the amount of hand annotation by moving to an online learning problem where the goal is to maintain a high classification accuracy, while recognizing that language, especially hate speech, evolves rapidly. These types of methods have focused on determining emerging words in the literature in order to improve training (emerging words).

With reports stating online hate speech has been increasing across the globe (UN), we feel the task of identifying toxic comments is becoming increasingly important. Our goal is to create a low-cost solution for increasing toxicity classification accuracy by using a very simple algorithm that will hopefully increase classification accuracy while limiting the need of hiring additional linguistic personnel/moderators. Our methods build on the aforementioned models, starting with a very simple method to determine linguistic differences between training and test sets, and then building on this with limited expert annotations. Our results suggest our methods do not enhance accuracy. In fact, all models perform the same. This suggests more exploration is needed to understand what makes certain examples harder to classify.

Dataset Construction

Our task uses the jigsaw\_toxicity\_pred (Huggingface). This dataset contains comments from Wikipedia that were hand labeled for being toxic, severely toxic, obscene, threatening, insulting, or related to identity hate. These categorizations are not mutually exclusive, meaning a comment could be both toxic and obscene. For our experiments, we focus solely on whether a comment was labelled toxic. We limit our experiment to two-class classification, as we believe noting whether something is toxic or not is the fundamental issue. While it would be nice to know the nature of the toxic comment, we don’t believe that this is as essential to companies trying to determine whether a comment should be removed for violating community standards.

Our dataset contains 159,571 examples in training and 63,978 examples in the testing dataset. 9.5% of the training examples are labelled toxic. Table 1 holds examples of a non-toxic and toxic example after pre-processing. Our pre-processing technique includes lower casing the text, removing all special characters (e.g., ascii characters), and numbers (see Github).

|  |  |
| --- | --- |
| **Label** | **Example Text** |
| Non-Toxic | congratulations from me as well, use the tools well |
| Toxic | cocksucker before you piss around on my work |

Table : Example text from dataset.

Methods

|  |  |
| --- | --- |
| **Log-odds** | **Vocabulary** |
| -41.19 | poop |
| -37.26 | dicks |
| -35.81 | gay |
| -34.45 | youfuck |
| -32.51 | n\*ggers |

Table 2: Most unique words in test dataset as compared to training dataset using log-odds ratios.

Our methods rely on using the test dataset to augment the training dataset in order to improve accuracy. We explore a combination of methods that include using a simple mathematical formula to determine which examples are more “unique” to the test versus train and simulating hand annotation. The former method uses log-odds ratios, while the latter relies on human expertise.

Our prior is that while the test dataset is pulled iid from the training dataset, there could be patterns in the test dataset that are not necessarily represented in training. Thus, we want to understand which examples are more representative of the testing dataset than the training dataset. In order to do this, we introduce log-odds ratios (Fighting Words). This method works by creating counts of all the words in the training and testing dataset, which are then converting these into proportions (equation 1). For words that exist in one but not the other, we use plus one smoothing. Next, we convert these into odds to determine the odds of word *w* appearing in dataset *i* (equation 2). For this paper, this translates to the odds of word *w* appearing in either the training or test set. This allows us to get a ratio of how likely a word is to appear in one dataset versus another by diving the odds for the training and the test dataset. We then log the ratio in order to make the numbers symmetric between the two groups (equation 3).

()

(2)

(3)

Using the final formula, we then apply this method to compare the language in the training and test datasets. In our case the most negative words are the words most unique to the test dataset. We pull these words and use them in our experiments to determine which examples we should pull from test and place into train. Our list of the top 5 words and their log-odds are below.

Our second method is more intuitive but relies on human expertise which can be costly. This method requires experts to evaluate the predicted test examples and determine which of the test examples are misclassified in order to use those examples to supplement the training dataset. We hypothesize that taking 20 random examples from test and placing them into train isn’t going to improve accuracy much because the 20 examples could be the easiest for the model to classify. Instead we believe that taking 20 hard examples, measured by whether the model classified them incorrectly, is likely more beneficial. If our hypothesis is true, as test datasets become larger and accuracy becomes higher, this process will become increasingly costly as it will take human experts longer to determine the incorrectly classified examples.

Experimental Set-up

Following the methods described above, we run the following 5 experiments. These experiments combine both the log-odds method to determine which examples we should augment our training examples with, as well as some simulations of hand annotation. We have designed our experiments to move from purely automated (i.e. lowest-cost) to a combination of automated and hand-annotated (i.e., higher-cost). The hope is the lower cost methods will be able to perform better than baseline without trading off too much from a more costly approach.

**BERT:** We use the Bidirectional Encoder Representations from Transformers model (BERT; Devlin et al., 2019) for our classification task. This is a transformer model that uses attention to understand sentences. We use the distilbert-base-uncased pre-trained model and fine-tune on jigsaw\_toxicity\_pred from Huggingface. We use the following hyperparameters: epochs 3, batch size 16, learning rate 2e-5, optimizer Adam, weight decay 0.01. We use distillBERT due to time constraints. We did run BERT as well and noticed no difference in accuracies for the baseline model.

**Experiment 0 (baseline):** We run distillBERT on the pre-processed dataset. We determine what the overall accuracy is for our model. We then pull the misclassified examples which will be used in experiments 3 and 4 to simulate hand annotation.

**Experiment 1:** We use log-odds ratios to determine the 20 most unique words in the testing dataset when compared to training. We then shuffle the training dataset and iterate through our list of words. For each word in our list, we pull one example from the test dataset that contains the word and then we remove the word from our list. We remove this final list of 20 examples from the testing dataset and place them into the training dataset. We rerun distillBERT on the new training dataset and evaluate using the new test dataset. We run a control model that uses the same testing dataset with the 20 examples removed, but with a training dataset augmented with 20 examples pulled from classic English novels (see Appendix). We add in 20 arbitrary training examples to ensure the model is not performing better just because there are new examples in the training dataset.

[Insert a table with the 20 most frequent words in the test dataset – all together and nouns]

**Experiment 2:** We repeat experiment 1, but we limit the 20 most unique words to now the 20 most unique nouns. We chose nouns because prior research has shown that nouns are the most likely to evolve over time (emerging words paper).

**Experiment 3:** We provide an expert annotator the test dataset with predicted labels. Their job is to determine the first 20 misclassified examples which are then taken from test and placed into train. We simulate this by taking a random 20 misclassified examples from test and placing them into the training dataset. We rerun distillBERT on the new training dataset and evaluate using the new test dataset. We then run the same control experiment outlined in experiment 1, with the same control training dataset but with the testing dataset used in this experiment.

**Experiment 4:** We add an additional layer to experiment 1 where now a hand annotator is given the list of 20 words. They are then given the test examples with the predicted labels. The dataset they are given is limited to only those examples containing the words from the list of 20. The dataset is sorted by these words. Their job is to read through the examples for each word and determine if the label is correct or not. For each word, once they determine the first incorrectly labelled comment for a word, they tag that comment and then move to the next word. The list of 20 examples they come up with are then removed from the test dataset and added to the training dataset. We simulate this experiment by pulling the first example for each word in our emerging word list from the misclassified examples from experiment 0. We pull these examples from test and place them into train. We rerun distillBERT on the new training dataset and evaluate using the new test dataset. We then run the same control experiment outlined in experiment 1, with the same control training dataset but with the testing dataset used in this experiment.

**Implementation and Evaluation:** We repeat experiments 1 through 4 using 50 examples as opposed to 20 to see if the number of training examples meaningfully affects accuracy. For each model we report the final accuracy of the model after evaluating it on the test dataset.

Results

Our results suggest that while all of the models achieved high accuracy, none of the edits we made improved the accuracy.

|  |  |  |
| --- | --- | --- |
| **Model** | **Accuracy**  **N=20** | **Accuracy**  **N=50** |
| E0: DistillBert | 94 |  |
| E1: Log-odds | 93 |  |
| E1: Control | 93 |  |
| E2: Log-odds Nouns | 93 |  |
| E2: Control | 93 |  |
| E3: Annotation | 93 |  |
| E3: Control | 93 |  |
| E4: Annotation using log-odds | 93 |  |
| E4: Control | 93 |  |

Table 2: Model Accuracies for each Experiment.

Discussion

While low-cost methods are an exciting research concept, our methods for augmenting the training dataset using information from the test dataset did not lead to more accurate classification. This result has led us to consider why we might be getting this result and further research questions to probe this task further.

We made a variety of assumptions due to time constraints for this project. For instance, we did not consider how the size of our dataset could necessitate more than 20 or 50 examples. In particular, there could be an optimal number of examples needed depending on the size of the training dataset.

Second, due to the length of time it took to run each model we were not able to explore different hyperparameters or adding additional epochs. It could be the case that our model wasn’t able to learn from the new examples.

Third, our dataset might not be the best for this task. Our method could be better for an online learning task, or a task where the language is dramatically different between the training and the test dataset. An initial analysis of the most unique words in the test versus the training dataset suggest that all of the words in the test dataset appear in the train dataset just with less frequency. Thus, it is likely the case that the examples we are supplementing the training dataset with are already well represented in the training dataset. Thus, we should consider another task, such as using Wikipedia comments from 2014 versus 2017, or some version of this that allows language to evolve more so that our method is able to pick up more unique differences.

Additionally, as previous researchers have shown, BERT models are doing very well and they seem to only mess up on cases that are hard even for human annotators. In particular, movie review sentiment classification tasks really only get wrong sarcastic reviews (cite), and sarcasm is a difficult topic to get at through only comments. Researchers have also shown that natural language inference tasks are difficult due to models relying on incorrect heuristics (cite). Both of these suggest that a very simple model, such as that suggested by us in this paper might not be enough to help with these final more difficult cases. In fact, we might need a slightly more advanced model that can identify certain elements of the text that could be leading to misclassification.

There are multiple issues for researchers to consider when thinking about augmenting a dataset. We chose our method because our dataset was pulled from one test period with a random training and test split. We believed there would be random differences between the training and test split, our method allowed for the algorithm to decide what those differences were. One could image an online learning setting, where this approach could actually work better. For instance, if the training and test datasets were fundamentally different (e.g., words appearing in one of the datasets were entirely absent from the other) then this algorithm could allow us to get more unique examples into training that could help. A combination of additional tweaks and new methods could help when developing low-cost methods to improve classification accuracy, but our results show that our method is not enough.

Acknowledgments

Thanks to Sasha Rush and our CS6741 classmates for helping develop our understanding of ML and NLP methods, as well as pushing us on our research questions.

References

Alfred. V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling, volume 1*. Prentice-Hall, Englewood Cliffs, NJ.

American Psychological Association. 1983. *Publications Manual.* American Psychological Association, Washington, DC.

<https://ai.facebook.com/blog/community-standards-report/>

https://www.ohchr.org/EN/NewsEvents/Pages/sr-minorities-report.aspx