IT3212

Project Report

Group 17

Word count: 4 908 | sections 1-5 excluding tables

Table of Contents

Problem definition

State-of-the-art

In a 2023 study by K. Shyrokykh et al., methods for classifying short texts are compared, in a small and imbalanced dataset of 5 750 tweets, on whether they are about climate change or not. The model that performed the best with a balanced dataset was convolutional neural network, with an accuracy of 97% and an F1-score of 0.97. The paper shows that simple supervised classifiers tend to perform just as well as advanced neural networks and therefore suggests that deep learning only should be used when there is a lot of high-quality data and extensive computational resources are available. [1]

A study done by Bonetti et al.(2023), compares a set of machine and deep learning methods to gauge which models are more efficient for toxicity detection in short form text messages. The methods explored in the study were logistic regression, random forest, and support vector model. Additionally the study compares these results to BERTweet, a variation of the popular BERT model which is pre-trained on English tweets. They combined 9 different open source datasets, and all the models presented accuracy and F1 scores in the 90% or above range. Although BERTweet achieved the highest scores overall, the associated computational overhead was deemed unwarranted in comparison to the small increase.[2]

ChatGPT has in the last year been on the forefront of the AI scene, being considered a state of the art language model. It's therefore natural to wonder about its application into detecting hateful speech. The paper written by Li et al.(2023) experimented on this, where ChatGPT was asked to identify whether a text was hateful, offensive, toxic (HOT) or a combination of this. The results were compared to real-human classification, and ChatGPT seemed to exhibit an accuracy of around 80%.[3]

A recent study done by Gasparetto et al.(2022), evaluated seven algorithms for text classification tasks, including classical methods like Naïve Bayes and linear support vector classifiers, as well as modern deep learning models such as FastText, Transformer-based language models (e.g., BERT, XLNet), and BiLSTM. The experiments were conducted on EnWiki-100 and RCV1-57 datasets, and revealed the dominance of Transformer-based language models, particularly BERT and XLNet, achieving state-of-the-art performance on various TC tasks. [4]

In a study by S. Afrifa and V. Varadarajan (2022), natural language processing and the machine learning techniques random forest and support vector machine were used to detect cyberbullying in a dataset consisting of 16 851 tweets. These two models managed to achieve accuracy scores of 98.5% and 90.5% respectively. [5]

[1] Shyrokykh, Girnyk, and Dellmuth, "Short Text Classification with Machine Learning in the Social Sciences."
[2] Bonetti et al., "Comparison between Machine Learning and Deep Learning Approaches for the Detection of Toxic Comments on Social Networks."
[3] Li et al., "'HOT' ChatGPT."
[4] Gasparetto et al., "A Survey on Text Classification Algorithms."
[5] Afrifa, "Cyberbullying Detection on Twitter Using Natural Language Processing and Machine Learning Techniques."

Problem statement

Real world problem

The problem we have decided to tackle is cyberbullying within text chat in video games. Online gaming has in the past 20 years exploded in popularity and has been a prevalent past-time activity amongst many youths. It is therefore quite concerning that most online gaming communities struggle with large amounts of toxicity, misogyny, and racism.

The cause of gaming toxicity is often attributed to the fact that users are protected by a screen and can act out without facing repercussions or relating to the person they are harming. Over time the normalization of toxicity in gaming culture has become a large attributor to the prevalence of hate speech in gaming communities.[6] Additionally in 2018 the World Health Organization classified a new disorder, aptly named "Gaming Disorder", which relates to an unhealthy addiction to video games. [7]

Traditionally game developers have used self-reporting systems to handle toxicity. In recent years, game developers and publishers have realized that AI can be leveraged to improve their cyberbullying prevention systems. As of the 30th of August 2023, Activision is employing ToxMod for their most recent Call of Duty titles, an AI system which is able to moderate live voice chats in game [8]. Even in the early stages the system shows great results, as 20% of players don't reoffend after receiving a warning. The players that do reoffend are banned or punished appropriately.

Activision's test run, coupled with market trends and advancements in AI technology, demonstrates the power of AI as a moderation tool. Its ability to comprehend natural language and understand intent behind sentences marks it as a solid solution for the prevalent cyberbullying problem. In summary: the real-world problem we want to tackle is the classification of cyberbullying in chat messages.

Machine learning problem

Based on the chosen dataset this is a binary classification problem. The binary classification algorithm categorizes new observations into one of two classes. Examples of this include: email spam detection, medical testing and in our case text classification. The two classes in binary classification are the normal state and the abnormal state. With our dataset the normal state will be text classified to not contain cyberbullying, with the label 0. The abnormal class is represented by the label 1 and will contain the text classified as cyberbullying. A part of the machine learning problem is to select the method of classification. [9]

Given the classification of a set of data, there are four combinations of the actual data category and the assigned category: true positives, true negatives, false positives and false negatives. In our text classification context, a true positive occurs when the model correctly identifies a piece of text as containing cyberbullying. For example, if the text is abusive or offensive and the model classifies it as such (label 1), it is considered a true positive. True negatives happen when the model accurately classifies text as not containing cyberbullying. That is, texts that are harmless are correctly identified and labeled as 0. A false positive in our scenario is when the model incorrectly labels a non-

[6] Fu, "A Look at Gaming Culture and Gaming Related Problems: From a Gamer's Perspective."
[7] Darvesh et al., "Exploring the Prevalence of Gaming Disorder and Internet Gaming Disorder."
[8] "Anti-Toxicity Progress Report – Voice Chat Moderation."
[9] Brownlee, "4 Types of Classification Tasks in Machine Learning."

cyberbullying text as cyberbullying. On the other hand, a false negative occurs when the model fails to identify actual cyberbullying, mistakenly classifying it as harmless text. [10]

Based on the combinations of data, different metrics can be used to measure performance of the classifier. For the comparisons of our models we used both accuracy and F1-score as performance metrics for the models.

Motivation

Tackling cyberbullying in online gaming chats is crucial for several reasons. It's essential for protecting players' mental health, as exposure to toxic behaviors can lead to anxiety and depression, especially among younger users. Ensuring an inclusive and welcoming environment in gaming communities is also vital, as cyberbullying currently inhibits diverse participation. With advancements in AI and machine learning, there's a growing responsibility for game developers to use these technologies effectively in combating online harassment and abuse. These efforts are not just about improving gaming experiences, but also about providing a safe arena for the players.

Pipelines

For all three pipelines we used the same three steps in preprocessing, feature extraction and feature selection. In the following paragraphs we will describe this process with the prediction of each model at the end.

Preprocessing

For preprocessing we applied common Natural Language Processing (NLP) techniques, as listed in Table 1.

Technique	**Description**
Removal of punctuations, special characters (#, %, & etc.)	Removing punctuations and special characters serves the purpose of reducing the dimensionality of our model and providing consistent tokenization of words.
Tokenization	Will be used to structure our words into a format that's better fit for most models. Standardizes the format of the data, ensuring consistency across the dataset.
Lower-case of words	Ensures that the words in different cases would be counted as the same, for example PIZZA and pizza should be counted as the same word. There is an argument to be made that capitalisation of words could have slightly different connotations. For example hate and HATE, where most people would consider HATE to have a lower sentiment score, but we will most likely choose to ignore this for our project.
Lemmatization	Helps the model recognize different variations of a word to be one and the same, for example “running” and “ran” become “run”. We chose to use this

[10] “Classification: True vs. False and Positive vs. Negative | Machine Learning.”

	over stemming, as it sometimes can be confusing for us humans to know what the stemmed word actually is meant to be in certain cases.
Sentiment analysis	Helps us gauge the overall sentiment of the sentence. The use of sentiment analysis is instrumental in determining the emotional tone of a message. In cyberbullying the emotional context (negative, positive or neutral) is often a key indicator of the nature of the communication.
Textual representation of emoticons	An approach to handling emoticons in the dataset. The approach replaces the emoticons with their corresponding textual representation. By doing so we preserve the emotional or contextual cues they provide, which might be important for sentiment analysis or understanding the tone of the message.

Table 1: Preprocessing techniques

One crucial aspect of our preprocessing strategy involved the decision to retain stop-words in the text data. Contrary to conventional practices, retaining stop-words yielded notable improvements in performance. We hypothesized that this could be attributed to the utilization of bi-grams, where the inclusion of stop-words facilitated the extraction of meaningful word pairs and enhanced the contextual understanding of the text. This strategic choice in preprocessing was especially beneficial in capturing nuanced relationships between words and contributed to a more robust representation of the data.

Including stop-words in conjunction with bi-grams aligns with the nature of our task, where preserving the intricacies of language is vital. This unconventional approach challenges the typical assumptions about stop-word removal and underscores the importance of adapting preprocessing steps to the specific characteristics of the dataset and the intricacies of the machine learning task. This exploration of stop-word retention adds a unique dimension to our preprocessing pipeline, contributing to the adaptability and effectiveness of our approach.

Feature extraction

To extract features from our data, we need to convert our textual data to numerical data. To achieve this, we used TF-IDF (term frequency-inverse document frequency) and sentiment analysis. TF-IDF calculates two different weights; term frequency (TF) and inverse document frequency (IDF). TF measures how often a term occurs in a document. Term frequency can also be adjusted for document length by dividing the count of occurrences by the number of words in the document. IDF measures how important a term is in the corpus. [11]

It is possible to use TF exclusively to extract features from textual data, which is called Bag of Words. This method would count the frequency of the words in a message. The features of a message would then simply be a vector with the frequency count of all its words. However, for our feature set, we opted to use IDF reweighting. The reason for this is that IDF can help minimize the impact of the more commonly appearing words in the English language that might not directly determine whether or not a message is considered cyberbullying. It also increases the impact of words with lower frequency. When it comes to classifying the messages, TF-IDF helps reduce the impact of more frequent and common words that are not important in order to determine the classification of the

[11] "Tf-Idf :: A Single-Page Tutorial - Information Retrieval and Text Mining."

message, while increasing the impact of less frequent and more uncommon words that are more helpful in determining if a message contains cyberbullying. TF-IDF is also computationally cheap and easy to calculate [12]. In addition, prior works on detection of cyberbullying in text have also found TF-IDF to be a useful tool for feature extraction, and it is considered to be a proven method for classifying text [13].

In addition to TF-IDF, we also use sentiment analysis to extract features from our data. Sentiment analysis is used to determine whether the emotional tone of a message is positive, negative or neutral [14]. For each of our documents, or messages, we calculate three values using sentiment analysis: a positive percentage, a negative percentage, and a neutral percentage. These three values show what percentage of the message is either positive, negative or neutral. As such, these three values add up to 100%. This can help discover any negative emotions or intentions, which can assist in discovering messages that contain cyberbullying.

The final feature set consists of n rows (equal to number of samples) of features. Each row includes the values from the sentiment analysis and all the TF-IDF weights for the message.

Feature selection

In our project, feature selection played a pivotal role in optimizing our machine learning models' performance while simultaneously addressing the issue of dimensionality reduction. Feature selection is a critical step in ensuring that our models are efficient, less prone to overfitting, and capable of delivering accurate results.

For our feature selection, we opted for the Chi-Square method. This method is particularly well-suited for text-based classification tasks, which aligns well with our project. It evaluates the correlation between individual terms (words or tokens) and the target label, assigning a value to each term that quantifies its relationship with the label. Essentially, the higher the Chi-Square score, the stronger the correlation between a term and the target label. Chi-Square not only measures these correlations but also allows us to rank the features. By scoring and ranking the terms, we could identify which terms were most relevant to the distinguishing of hateful and non-hateful speech. This ability to rank features helped us prioritize the most informative terms for our models.

For example the feature(word/token) “hate”, suppose the word appears frequently in texts labeled as cyberbullying and rarely in text labledes as non-hatful. The Chi-Square will calculate the correlation between the presence of the word and the likelihood of the text being cyberbullying. Given its high Chi-Square score, the word “hate” would be a significant predictor of cyberbullying, thus included in the model. In contrast the feature(word/toke) “the”, is a common English word and is likely to appear frequently across both cyberbullying and non-hateful chat messages, showing no specific pattern or correlation with the target labels. The Chi-Squared test would likely provide a low score for “the”, consequently it would be excluded from the model.

One of the primary challenges in feature selection is determining the appropriate value of k, which represents the number of features we choose to retain after selection. A well-chosen k-value is crucial

[12] “Understanding TF-IDF for Machine Learning.”
[13] Muneer and Fati, “A Comparative Analysis of Machine Learning Techniques for Cyberbullying Detection on Twitter.”
[14] “What Is Sentiment Analysis?”

for balancing the preservation of essential information against the risk of overfitting. In our case, we set the k-value to 300 after thorough experimentation and analysis. Selecting the right k-value is a balancing act. A smaller k-value might result in a significant loss of data and could lead to a lack of critical information, while a larger k-value may increase the risk of overfitting. Our choice of 300 was the result of extensive model testing and analysis. It allowed us to strike the right balance between maintaining sufficient features and mitigating overfitting.

Another crucial refinement was introduced to address the inherent challenges associated with the dataset's sentiment imbalance, particularly the prevalence of negative sentiment in the dataset. As part of our feature engineering strategies, we opted to incorporate sentiment analysis into our model. However, recognizing that negative sentiment did not strongly correlate with cyberbullying instances and was contributing to a notable trend of false negatives, we strategically performed feature selection. Specifically, we refined our feature set by including only the positive sentiment values derived from sentiment analysis. This deliberate exclusion of negative sentiment values aimed to realign the model's learning process with the nuances of cyberbullying, where positive sentiment, rather than negative, was a more indicative factor. Such feature selection not only contributed to a more balanced confusion matrix but also underscored the significance of thoughtful feature engineering in addressing the unique challenges posed by the dataset.

Prediction

To guide our model selection step, we performed Principal Component Analysis (PCA) on our TF-IDF transformed data. PCA is a dimensionality reduction technique that allows us to visualize and understand the distribution and variance of data in a lower dimensional space.

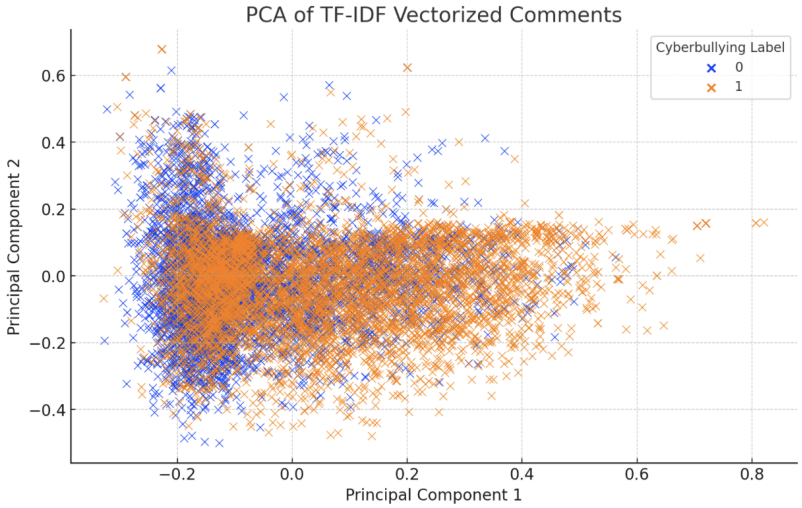

Figure 1: PCA plot of TF-IDF data.

The PCA plot, shown in Figure 1, provided several valuable insights into our data. There seems to be a significant amount of overlap between the data points labeled as ‘0’ and ‘1’. This suggests that the two categories are not linearly separable, which led us to select models capable of capturing complex relationships, for our pipelines. Additionally, the absence of distinct clusters in the plot indicates that there are no clear boundaries separating the two categories, this led us to choose Convolutional Neural Networks (CNN) known for their strong pattern recognition capabilities.

Support Vector Machine

TF-IDF vectorization results in a high-dimensional feature space, where each word and bi-gram becomes a feature. SVM is capable of handling high-dimensional feature spaces effectively, making it a suitable choice. It is also flexible in handling a combination of features, including the TF-IDF vectors and sentiment features. The model can effectively work with hybrid feature sets, and the inclusion of sentiment analysis can enhance its ability to detect cyberbullying text, which often exhibits distinct emotional tones. SVM seeks to find the hyperplane that maximizes the margin between classes, which results in a robust decision boundary that generalizes well to new data. This property is particularly valuable in binary classification tasks where the goal is to minimize misclassifications and achieve high accuracy.

We tuned the hyperparameters using grid search, which trains the model with several different combinations of hyperparameters, and evaluates the best one. Cross-validation is typically used, to ensure robustness. We used grid search because it automates the hyperparameter tuning process, which can be both challenging and time consuming to do manually, especially when the feature space is high dimensional and there is low interpretability. The hyperparameters that were tuned were: C, kernel selection and gamma. The most important hyperparameter to tune for our model turned out to be the C parameter. C represents the trade-off between achieving a low training error and a low testing error. Smaller values of C impose stronger regularization, which can help prevent overfitting. We explored the values 0.1, 1.0 and 10, ending up with 0.1. Small values for C lead to a wider margin, meaning that the model may allow some training points to be misclassified, this can help create a generalized model and therefore prevent overfitting. Large values for C lead to a strict model with a narrow margin, this can lead to overfitting if the data is noisy and not linearly separable. Since we initially deemed the data to be quite noisy with quite a lot of misclassification, it makes sense that the grid search favored a more generalized model with a wider margin. In terms of kernel selection and gamma, we used a “rbf” (Radial Basis Function) kernel and a gamma value of “scale” as this was evaluated by the grid search as the best values. The fact that SVM performed best with the “rbf” kernel rather than a “linear” one, confirms our initial observation from the PCA plot in Figure 1.

Ensemble learning - stacking

Ensemble learning involves combining the strengths of multiple base models to create a meta-model that makes more accurate predictions. For our ensemble we chose SVM, Naïve Bayes, k-Nearest Neighbours (KNN) and Decision Tree Classifier as our base models and Linear Regression as our meta model. Linear Regression was chosen as our meta model for its interpretability, since model transparency is a critical factor in the context of cyberbullying detection.

There are several benefits of stacking, especially for a binary text classification problem. Text data often contains nuanced language and subtle contextual cues. Stacking different models, each with its strengths, can enhance the ensemble's ability to understand and interpret these nuances. For example, KNN can bring in sensitivity to semantic similarities between instances, contributing to better contextual understanding. Whereas SVM is great at discerning patterns and capturing explicit relationships within the text.

Both random and grid search hyperparameter tuning was tested in order to optimize and increase the accuracy of the ensemble model. However, grid search was proven to be excessively time-consuming on our dataset. As a method of decreasing the execution time, we attempted to only run the tuning on a subset of data. This led to severe overfitting on the subset, and produced poor predictions. Therefore

random search was chosen for the pipeline. The number of iterations done by the random search was set to 20, a fair balance between time to execute and quality of the results.

The main goal of the hyperparameter selection was to increase performance by finding a fair balance between over- and underfitting to the dataset. Table 1 lists the parameters chosen for each model in addition to a short description of how they modify the model and why they were selected.

Model	**Parameter**	**Description**
Multinomial Naive Bayes	alpha	Determines the laplace smoothing of the algorithm. A wide range has been explored in order to balance over- and underfitting.
Support Vector Machine	C	Determines regularization strength, i.e. the margin used by the support vectors.
	kernel	The kernel in the SVC is the component which groups data points. A “linear” kernel will be better suited for data that is linearly separable, whereas the polynomial “rbf” kernel is better suited for complex data.
Decision Tree Classifier	max_depth	Determines the maximum depth of the decision tree’s branches. A value of None will allow the tree to grow until samples are fewer than min_samples_split.
	min_samples_split	Minimum number of samples until the node is split. Along with max_depth this parameter has been tuned to avoid overfitting to noise in the dataset.
K-Nearest Neighbors	n_neighbors	Number of neighbors to use per sample. Larger values can lead to underfitting, and smaller values can lead to overfitting.
	weights	Whether to apply an equal weight to every node in the neighborhood, or to weigh them by distance between them.
Logistic Regression	C	Strength of the regularization. Smaller values increase regularization and prevent overfitting, but might harm accuracy.
	solver	The optimization algorithm to use. Generally liblinear is better for smaller datasets, and lbfgs is better for larger datasets.

Table 2: Hyperparameters which were chosen for tuning.

The hyperparameter tuning is implemented using the RandomizedSearchCV model. Before the search, sets of possible values for each parameter were selected manually. Afterwards the model was fitted to our dataset, where it runs cross-validation on the parameters and determines the best value for

each parameter. The possible values for each parameter alongside the result of the randomized search is listed in table 3.

Model	Parameter	Possible Values	Best value
Multinomial Naive Bayes	alpha	0.1, 0.5, 1	0.1
Support Vector Machine	C	0.1, 1, 10	0.1
	kernel	linear, rbf	rbf
Decision Tree Classifier	max_depth	None, 10, 20	None
	min_samples_split	2, 5, 10	2
K-Nearest Neighbors	n_neighbors	3, 5, 7	3
	weights	uniform, distance	distance
Logistic Regression	C	0.1, 1, 10	1
	solver	liblinear, lbfgs	liblinear

Table 3: Values tested for each hyperparameter, and the value which was selected by the random search.

Convolutional Neural Network

A Convolution Neural Network (CNN) is used to extract features from grid-like matrix datasets[15], and can therefore be used with almost any type of data that can be represented in this format. By using a one-dimensional CNN, we can use the feature vectors containing TF-IDF scores and sentiment analysis results as input for the model.
After trying multiple different model architectures to find out which resulted with the highest accuracy score, we ended up on the model shown in Table 4.

Model architecture
Convolutional layer with 32 filters, kernel size of 5 and same padding, ReLu activation func.
Max-pool layer with pool size of 2 and stride of 2
Dropout with rate of 0.5
Flatten layer

[15] "Introduction to Convolution Neural Network."

Dense with output space of 1, sigmoid activation function

Table 4: CNN model architecture

Adding more layers only seemed to worsen the result.
We tuned the hyperparameters for the model primarily by trying different values and observing how they affected the performance of the model, specifically the accuracy on the test set. The tuned parameters are presented in Table 5.

	Hyperparameter
Convolution layer	Filter size: We settled on a filter size of 5 the
	Stride: We let stride stay at 1 as increasing it did not seem to positively affect the accuracy of the model.
	Padding: This was set to "same", resulting in padding with zeros evenly to the left/right or up/down to ensure that we get an output with the same width and height as the input.
	Activation: ReLu was found to produce the best results.
Model	Learning rate: We used a learning rate of 0.001 with the Adam optimizer as this resulted in the highest accuracy on the test set.
	Loss function: We used binary-cross entropy as it's generally recommended for binary classification tasks
	Epochs: The model was trained for 15 epochs,though the accuracy seemed to stagnate around 10 and only marginally increasing after

Table 5: Hyperparameters tuned for CNN

One of the issues we initially ran into was overfitting. The accuracy during training was much higher than the accuracy on the test set. To combat this, we added regularization, which is a way to add penalties to the model during training. We opted for L2 regularization as it seemed to produce the best results. L2 helps spread the influence of a feature to multiple other features, which can help prevent overfitting when there are correlations between different features[16].
We also added a pooling layer (max-pooling) and a dropout after the convolutional layer. Pooling layers reduce the dimensions of the feature maps which in turn makes the model more robust to changes to the position of features in the input matrix[17]. It also helps speed up computation time as it reduces the number of parameters. Dropout works to avoid overfitting by randomly nullifying the contribution of some neurons towards the next layer which helps reduce how much the first batch of training samples influence the learning[18].

[16] Otten, "L1 And L2 Regularization Explained, When To Use Them & Practical How To Examples."
[17] "CNN | Introduction to Pooling Layer."
[18] baeldung, "How ReLU and Dropout Layers Work in CNNs | Baeldung on Computer Science."

Comparison of performance

The performance of the three models are presented in table 6, both the accuracy and F1-score are used as performance metrics.

Model	Accuracy	F1-Score
Support Vector Machine (SVM)	73.24%	0,71
Ensemble Learning-stacking	72,97%	0,71
CNN	73,51%	0,73

Table 6: Accuracy and F1-score for the models SVM, Ensemble learning - stacking and CNN

We see that the pipeline utilizing CNN as its prediction model performed the best on both metrics among the three models. The models' ability to capture sequential patterns and hierarchical features likely contributed to its superior performance. The SVM pipeline also achieved a reasonably good accuracy and F1-score and is known for its effectiveness in handling binary classification tasks. SVMs performance suggests that it is competitive and could be suitable for datasets with linear or less complex relationships. The stacking pipeline achieved the lowest accuracy among the three models, in addition to having the longest runtime. The fact that our stacking pipeline performed worse than SVM, even though it included SVM as one of its base models, means that we were not able to get any added value from our ensemble of diverse base models. In hindsight we regret not optimizing each individual base model before incorporating them into the stacking ensemble. By not doing this we lacked understanding of how each model performed on the dataset, in addition this made it very difficult and time consuming to tune the ensemble.

Overall, we interestingly noticed that all the three pipelines achieve around the same accuracy and F1-score. This suggests that our models may be facing similar challenges or limitations. For instance, one reason might be that there is not enough discriminative information in the extracted feature set for the models to capture subtle nuances for accurate classification. While another reason might be due to the data quality being somewhat compromised, which we have discovered through finding numerous misclassifications.

Conclusively, to break through this performance plateau, it might be beneficial to revisit aspects of the data preprocessing and feature engineering, and also consider using different approaches for the different pipelines.

Comparison to state-of-the-art

To understand why our models did not achieve state-of-the-art performance, we tried analyzing our model's misclassifications. In all of our models we observed a notable trend towards false negatives—instances where the model predicted "not-cyberbullying" when it was, in fact, cyberbullying. This inclination was evident in our ensemble stacking model's confusion matrix, which revealed that 68% of our misclassifications were false negatives. We attribute this pattern to the prevalence of negative sentiment throughout our dataset. This can be visualized by a word cloud of the data labeled as non-cyberbullying, as illustrated in Figure 2.

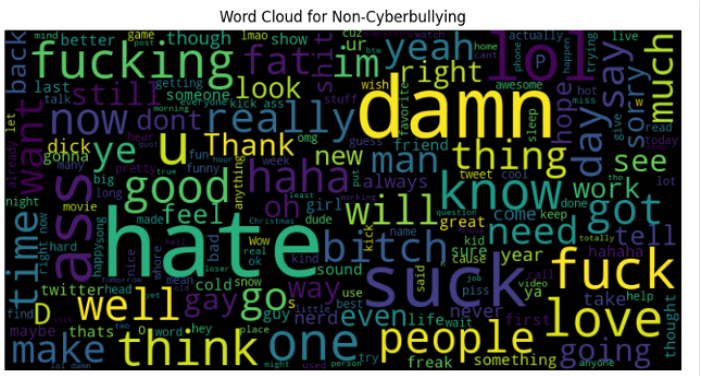

Figure 2: Word cloud for non-cyberbullying.

The word cloud showcases the abundance of negatively loaded words such as "hate". Consequently, our models become trained on data where negative sentiment, often associated with words like "hate", is frequently interpreted as "not cyberbullying", see figure 3. This likely contributes to the higher frequency of false negatives, posing a challenge for our models to achieve accuracy comparable to state-of-the-art benchmarks.

As mentioned earlier, in our feature selection step, we made an effort to address this misclassification imbalance by modifying our feature set to only include positive sentiment values, recognizing that negative sentiment does not strongly correlate with cyberbullying. This adjustment significantly balanced our confusion matrix, offering a more even distribution of predictions by giving a 9% decrease in false negatives, while maintaining the same accuracy.

Text: lol whore bath funny
True Label: Toxic
Predicted Label: Non-Toxic
Weighted prediction: [0.57928145 0.42197162]
The word lol appeared 595 times. 379 times it had label Non-Toxic, 216 times it had label Toxic
The word whore appeared 84 times. 33 times it had label Non-Toxic, 51 times it had label Toxic
The word bath appeared 4 times. 2 times it had label Non-Toxic, 2 times it had label Toxic
The word funny appeared 76 times. 40 times it had label Non-Toxic, 36 times it had label Toxic

Figure 3: Example of a predicted non-cyberbullying message

Figure 3 shows an example of misclassification in our CNN model, where the original dataset classifies it as cyberbullying (Toxic), and our model as not cyberbullying (Non-toxic). One possible reason for this is that the word “whore” is frequently used in what’s qualified as Non-Toxic speech. In the dataset, the word “whore” appeared 84 times, yet 33 of those times it was included in text labeled as Non-Toxic. This could contribute to the model’s assumption of this being Non-Toxic.

We discussed ways to approach this, however it seemed like a difficult problem to solve. One solution could be to gather a list of words normally deemed toxic, and that were common in FN examples, and give them a weight to try to “boost” their negative sentiment score. However, even if we managed to do this resource-intensive task, it would probably not increase our performance, seeing as the original dataset quality was compromised by having quite a lot of misclassifications.

Ultimately, it is clear that achieving state-of-the-art performance in our cyberbullying classification task is hindered by the inherent challenges posed by the negative nature of the dataset. Distinguishing genuine instances of cyberbullying from 'regular' negative speech proves to be a difficult task, given

the commonalities between the two. For instance, expressions of frustration or disagreement, often present in both categories, contribute to the complexity of accurate classification. To overcome this challenge, it might be beneficial to explore more nuanced features or contextual clues that can help differentiate between harmless negativity and potentially harmful cyberbullying.

Discussion

Machine learning solution

For our machine learning solution, we've encountered several challenges in tackling the text message dataset. The first of which were feature extraction and selection, as the dataset initially only consisted of two dimensions, the text and the toxicity label. As such we had to extrapolate and create additional dimensions, which we decided to do using TF-IDF with ngrams.

One of our main takeaways is that we've identified a ceiling to the performance of all our pipelines. This ceiling is considerably lower than the ones presented in our state-of-the-art analysis, and lower than what is desired to actually solve the problem. Therefore it is reasonable to assume that there is still much room for our pipelines to improve.

One hypothesis for the performance ceiling is related to the quality of our dataset. Many of the messages presented in the dataset are seemingly incorrectly labeled. As an example the following message: "man that rly sucks. I for 1 am positive that all will work out. You're a bright dude... pretty cool 2 if I don't say so myself", has been flagged as toxic. Due to the usage of TF-IDF, this could cause our model to falsely infer a term such as "bright" with toxicity if it is less prevalent in the rest of the dataset. To avoid this the balance between over- and underfitting to the dataset has to be very precise.

To solve the aforementioned issue, future iterations of the models and pipelines should consider data collection from multiple sources in order to increase the variance in knowledge, such as done by Bonetti et al. (2023).

Real world solution

For our real-world solution, we propose a cyberbullying detection program utilizing our trained ML model. This program examines individual text messages, categorizing them as cyberbullying or not. It is designed for applications seeking to enhance the safety of their chat functionalities, allowing them to identify and manage cyberbullying instances by either filtering messages or implementing appropriate penalties.

However, our ML model is not perfect, leading to occasional misclassifications. To address this, we would inform our customers about the potential for innocent messengers to be penalized if the model's limitations are not considered. Recognizing the impact on user experience, we recommend an additional solution—accumulating a behavior score based on several chat messages. If a user's score significantly leans towards a cyberbullying nature, they may be automatically penalized or subjected to manual review.

With the ML model and these suggestions, application developers have the flexibility to determine the desired level of strictness or accuracy for their moderation. This allows them to strike a balance between avoiding unjust penalties for users and incorporating manual intervention as needed.

Group reflection

We decided early on that we wanted feedback on every part of the assignments, so that we had the best base for development and improving our group assignments as we went on. For the most part the feedback was clear and a useful tool for improving our assignments, but the received feedback in assignment 3 was more confusing than guiding. We might have misinterpreted the assignment due to a confusing assignment description.

Individual contributions and reflections

Ole

Contributions during the course:

- Assignment 1: Machine learning problem
- Assignment 2: Comparison of the models
- Assignment 3: CNN
- Final report: Problem definition, SVM pipeline, CNN pipeline

During the course all group members participated in group meetings ahead of each assignment. In each meeting we had an open discussion on how to tackle the assignment and distributed tasks among the members. This was a good habit, enabling me and the others to work on the assignments and the project independently. For the last report and assignment me and Joel worked together, which I experienced as a nice change in work environment.

Joel

Contributions during the course:

- Assignment 1: Common sources of noise
- Assignment 2: Feature extraction
- Assignment 3: CNN
- Final report: Problem definition, SVM pipeline, CNN pipeline

Overall, I felt like the group work went very well. The meetings we held before the assignments to discuss and distribute work were useful in ensuring that we were all on the same page in terms of what needed to be done and which direction we were taking. During some of the assignments, we also split into smaller teams to collaboratively work on certain tasks, which I thought was helpful as it allowed us to share ideas and come up with solutions more effectively.

Stian

Contributions during the course:

- Assignment 1: Real world problem
- Assignment 2: Implementation of preprocessing, Pipeline design
- Assignment 3: Selecting and tuning of hyperparameters, Results
- Final report: Research paper summary, Machine learning solution

I am quite happy with the course. Applying machine learning and pipeline design through a project has been very insightful and helped me gain a better understanding of how these techniques are used in practice. The work process of the group has also been nice, we met up to synchronize before starting work on each assignment, and followed up with individual or pair work for assigned

segments. My only gripe with the course is that the descriptions for each assignment were written too concisely, and that the actual requirements have been difficult to find/understand.

Knut

Contributions during the course:

- Assignment 1: Preprocessing
- Assignment 2: Implementation of preprocessing, feature selection and Naïve Bayes model
- Assignment 3: Implementation of LSTM model
- Final report: Research paper summary, Tuning of CNN model, analysis of misclassifications for CNN model

The course was interesting, and I feel I learnt a lot of practical knowledge about data-driven programming. I already have some experience with NLP, but it was cool to learn about other branches of modeling. In addition I'm a big fan of this course being mainly a project for evaluation, as it lets us use theory in practice. The group itself was really good and I felt we managed to work efficiently and divide the tasks in a fair manner. The only slightly negative point was that assignment descriptions were at times a bit unclear, especially in assignment 3 where there could have been put more emphasis on the data-analysis part (even though this has been talked about in some lectures).

Truls

Contributions during the course:

- Assignment 1: Real world solution
- Assignment 2: Implementation of preprocessing
- Assignment 3: LSTM description and comparison
- Final report: Real world solution, Research paper summary

This course was engaging and packed with ML knowledge and methods. At times it felt a bit much for me, but that is alright in a subject with such a hands-on approach, where the lecture material seemed, and worked, like a base for freetime/project work. I feel very satisfied with what I have learned about data: how to think about, treat it and train ML models on it. Something negative to mention is that I feel like I am in the dark on parts of the subject. I never found the time to dive into many aspects outside of what was relevant to the project. Perhaps this is expected, but feels like I have missed out a little.

David

Contributions during the course:

- Assignment 1: Machine learning solution
- Assignment 2: Implementation of modeling methods
- Assignment 3: Implementation of advanced pipeline
- Final report:
    - Research paper summary
    - Ensemble pipeline
    - Comparison of performance
    - Comparison to state-of-the-art

I found the course to be very interesting and I learned a lot about a domain that was entirely new. I liked how the lectures were very information dense, and since I knew that I wouldn't have to

memorize all the information for an exam, I was able to focus on overall understanding rather than memorizing. The projects were also a nice way to apply the theory and get the hands dirty. The only critique might be that the assignment descriptions could be a bit more clear, as they can be a bit vague when one is new to the field such as I was.

Citations

Gasparetto, A., Marcuzzo, M., Zangari, A., & Albarelli, A. (2022). A Survey on Text Classification Algorithms: From Text to Predictions. Information, 13(2), 83. https://doi.org/10.3390/info13020083

https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative

Afrifa, Stephen. "Cyberbullying Detection on Twitter Using Natural Language Processing and Machine Learning Techniques." *International Journal of Innovative Technology and Interdisciplinary Sciences* 5, no. 4 (December 27, 2022): 1069–80. https://doi.org/10.15157/IJITIS.2022.5.4.1069-1080.

"Anti-Toxicity Progress Report – Voice Chat Moderation." Accessed November 30, 2023. https://www.callofduty.com/blog/2023/08/call-of-duty-modern-warfare-warzone-anti-toxicity-progress-report.

baeldung. "How ReLU and Dropout Layers Work in CNNs | Baeldung on Computer Science," May 30, 2020. https://www.baeldung.com/cs/ml-relu-dropout-layers.

Bonetti, Andrea, Marcelino Martinez-Sober, Julio C. Torres, Jose M. Vega, Sebastien Pellerin, and Joan Vila-Frances. "Comparison between Machine Learning and Deep Learning Approaches for the Detection of Toxic Comments on Social Networks." *APPLIED SCIENCES-BASEL* 13, no. 10 (May 14, 2023): 6038. https://doi.org/10.3390/app13106038.

Brownlee, Jason. "4 Types of Classification Tasks in Machine Learning." *MachineLearningMastery.Com* (blog), April 7, 2020. https://machinelearningmastery.com/types-of-classification-in-machine-learning/.

Capital One. "Understanding TF-IDF for Machine Learning." Accessed October 25, 2023. https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/.

Darvesh, Nazia, Amruta Radhakrishnan, Chantelle C. Lachance, Vera Nincic, Jane P. Sharpe, Marco Ghassemi, Sharon E. Straus, and Andrea C. Tricco. "Exploring the Prevalence of Gaming Disorder and Internet Gaming Disorder: A Rapid Scoping Review." *Systematic Reviews* 9, no. 1 (April 2, 2020): 68. https://doi.org/10.1186/s13643-020-01329-2.

Fu, Daniel. "A Look at Gaming Culture and Gaming Related Problems: From a Gamer's Perspective," n.d.

Gasparetto, Andrea, Matteo Marcuzzo, Alessandro Zangari, and Andrea Albarelli. "A Survey on Text Classification Algorithms: From Text to Predictions." *Information* 13, no. 2 (February 2022): 83. https://doi.org/10.3390/info13020083.

GeeksforGeeks. "CNN | Introduction to Pooling Layer," August 5, 2019. https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/.

GeeksforGeeks. "Introduction to Convolution Neural Network," August 21, 2017. https://www.geeksforgeeks.org/introduction-convolution-neural-network/.

GeeksforGeeks. "What Is Sentiment Analysis?," July 10, 2020. https://www.geeksforgeeks.org/what-is-sentiment-analysis/.

Google for Developers. "Classification: True vs. False and Positive vs. Negative | Machine Learning." Accessed November 30, 2023. https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative.

Li, Lingyao, Lizhou Fan, Shubham Atreja, and Libby Hemphill. "'HOT' ChatGPT: The Promise of ChatGPT in Detecting and Discriminating Hateful, Offensive, and Toxic Comments on Social Media." arXiv, April 20, 2023. https://doi.org/10.48550/arXiv.2304.10619.

Muneer, Amgad, and Suliman Mohamed Fati. "A Comparative Analysis of Machine Learning Techniques for Cyberbullying Detection on Twitter." *Future Internet* 12, no. 11 (November 2020): 187. https://doi.org/10.3390/fi12110187.

Otten, Neri Van. "L1 And L2 Regularization Explained, When To Use Them & Practical How To Examples." Spot Intelligence, May 26, 2023. https://spotintelligence.com/2023/05/26/l1-l2-regularization/.

Shyrokykh, Karina, Max Girnyk, and Lisa Dellmuth. "Short Text Classification with Machine Learning in the Social Sciences: The Case of Climate Change on Twitter." *PloS One* 18, no. 9 (2023): e0290762–e0290762. https://doi.org/10.1371/journal.pone.0290762.

"Tf-Idf :: A Single-Page Tutorial - Information Retrieval and Text Mining." Accessed October 25, 2023. https://tfidf.com/.