Detect LLMs Generated Text

Daohang Tong

Computer Science
College of Natural Science
The University of Texas at Austin

ABSTRACT

The fast progress of large language models (LLM) has brought attention and scrutiny from various parts of the society. As with any powerful technology, it can be used for good and malicious intents. As the danger of such malicious use of LLM become more concerning, the effective way of detecting if a given excerpt of text is written by human or LLM becomes a necessary and important task. We formalized this problem as a binary classification problem and used area under the receiver operating characteristic curve (AUC-ROC) to measure our models' performance. We explored a wide range of models, including Logistic Regression, Stochastic Gradient Decent Training on Linear Models, Multinomial Naive Bayes, Gradient Boosting, and Transformer models. We also searched hyperparameter space to tune each model and combined individual models with an Ensemble model. To overcome the large vocabulary size issue introduced by typos, we built a custom tokenizer which turned out to be very effective and improved the performance of many models.

1. INTRODUCTION & RESEARCH BACKGROUND

As LLM becomes more powerful at reasoning and mimicking human's writing, at various tasks and professional workflows such as news generation, code generation (Zheng, 2023) (Copilot, 2021) and advertising creation (Murakami, 2023). Not only do LLM possess great capabilities, thanks to the rapid adoption and commercialization of LLM models like ChatGPT (OpenAI, 2023), the effect of LLM is profound and ubiquitous in various sectors in the society including Education, Law and Research. ChatGPT has fundamentally transformed the way of traditional education and homework system (Susnjak, 2022) as 89% of students admit to using ChatGPT for homework (Chris, 2023). As with any powerful technology, the power of LLM may be exploited by bad actors to spread disinformation (Pagnoni, 2022) (Lin, 2022) and perform online fraudulent schemes (Weidinger, 2021)(Ayoobi, 2023) and social media spams (Mirsky, 2023). To make matter worse, even for the legitimate use cases, the inherent flaw of LLM make them prone to "hallucination" (Ji, 2023) and thus can spread disinformation. OpenAI

rightfully pointed that reliability is important, when using the result from its models. (OpenAI, 2023). Moreover, some argue that LLM encourage the act of plagiarism by not mentioning the source (Lee, 2023).

As a result, it has never been more important to reliably detect the content generated by LLM. There are several types of detector methods such as watermark, zero-shot and LLMs as detectors. Watermarking techniques employs a set of "green" tokens and "red" tokens and encourages the LLM to pick tokens from "green" token during sampling, which effectively applied specific word or phrase pattern to the generated text. To detect such water mark, green and red token are combined to perform statistics and calculate the p-value. However, it has been shown that watermarking is susceptible to paraphrasing attacks. (Sadasivan, 2023) showed that when a light paraphraser is applied on top of the generative text model, it can effectively break the watermark.

Zero-shot method means the ability detect AI generated text without additional training on the test dataset distribution. Effectively, a general pre-trained model that is trained on a variety of styles of AI and human generated text may be effective in the zero-shot setting. Recent work like DetectGPT (Mitchell, 2023) demonstrates the effectiveness of such zero-shot models across a wide range of domains such as code and new article generation. In this paper, we will focus on the zero-shot method since it is easy to be generalized and does not make specialized assumption to the distribution of the LLM generated text.

2. DATA

To effectively train the models, our training data sets includes both essays from human and LLMs. The human written part is found in the following research: “The PERSUADE 2.0 corpus comprises over 25,000 argumentative essays produced by 6th-12th grade students in the United States for 15 prompts on two writing tasks: independent and source-based writing.” (Crossley, 2023). The selection of these essays ensures the human writing samples are well generalized across demographic, writing subjects, formats, etc. Such robustness will help us deriving the key difference between the two kinds of essays. Compared to handwritten essays, LLMs generated essays are much easier to collect. There are a total of 3000 essays which are generated by different LLMs such as ChatGPT, LLaMA, Falcon, Claude and PaLM (Kleczek, 2023). This approach allows us to blend a variety of LLMs in order to provide us more realistic and well-balanced

examples of LLMs generated essays. To effectively categorize the essays, our model will train on part of this data set and test on the rest.

In this research, our goal is to find ways to distinguish whether an essay is created by human or LLMs. As discussed earlier, our model will train and test on the data sets which includes hand-written essays and generated ones. Thus, the source data set includes two key columns which are source text and label of whether it is generated. Label "1" means the essay is generated by LLMs, while "0" represents human written essays. For the purpose of robustness and diversity, our data set includes essays on fifteen different topics and assort human writers as well as different large language models. The essay topics are shown in the column "prompt_name". The specific large language model is indicated in the column "source". The whole datasets include 44867 entries, while 27371 of them are written by human and 17497 are not. (Kleczek, 2023). After some basic data cleaning and preparation, we perform some data exploratory analysis.

3.1 Handwritten Essays

Essays which are written by human are collected in the data set called "Persuade 2.0 corpus" (Crossley, 2023). The writing samples were collected to better understand students' ability of writing persuasive articles. The researchers collect vast corpus to do a quantitative analysis on them in order to help students improve their argumentation. The corpus provides realistic source of human essays. The original data includes over 25,000 essays created by 6th to 12th grade students in the USA. The students need to perform one of the two kinds of writing on the 15 topics. Their task is to either write the essay solely by themselves or produce an essay based on external source. The strategy allows us to capture a broad spectrum of writing patterns and styles. With the massiveness of the corpus, the data is a well representation of human writing articles.

For a more comprehensive understanding of the data, we will begin with some exploratory data analysis (EDA) on our data which will show three main aspects of our dataset.

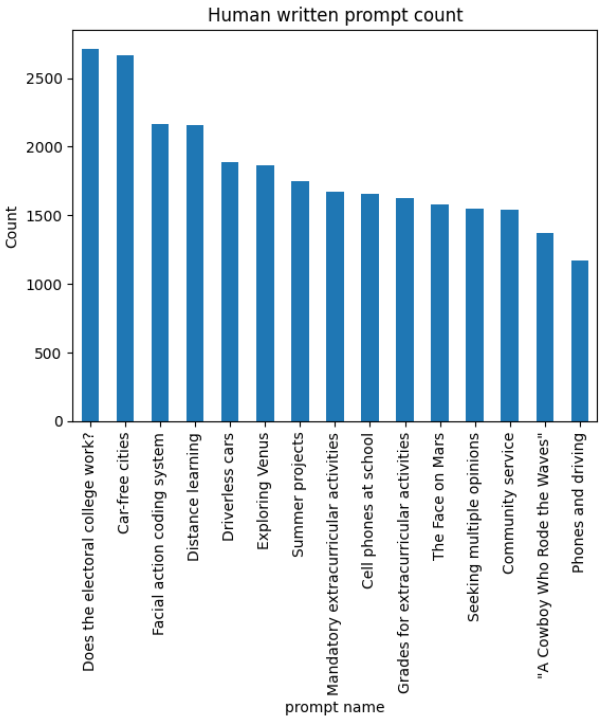
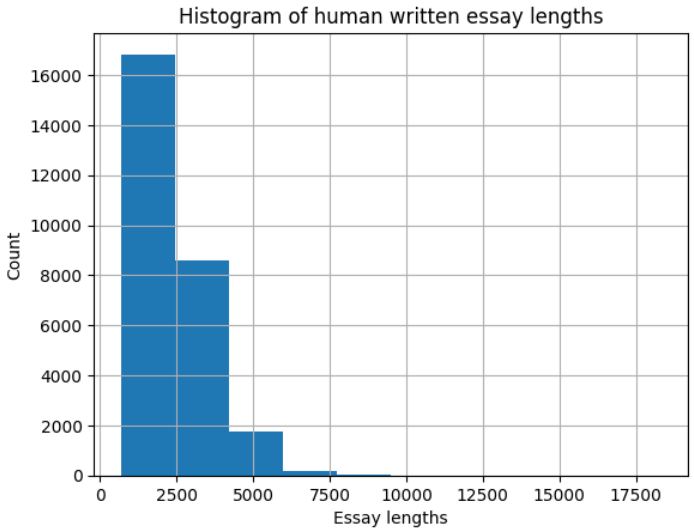


Figure 1. Human written prompt count

First, we categorize the texts by their prompt names and count the essays under each name. In the Figure 1, the histogram shows the count of each prompt. The top three topics written by human are “Does the electoral college work”, “Car-free cities” and “Facial action coding system”.

Figure 2. Histogram of human written essay lengths



count	27371.000000
mean	2348.503891
std	1124.489852
min	691.000000
25%	1499.500000
50%	2132.000000
75%	2929.000000
max	18322.000000

Table 1. Descriptive statistics of human written essay lengths

In terms of the lengths of the essays, from the Table 1 and Figure 2, we can see the medium is 2132 words and the mean is 2348, which means most essays are around two thousand words. We also notice there are many outliers from the Figure 2, meaning there are some essays with more words.

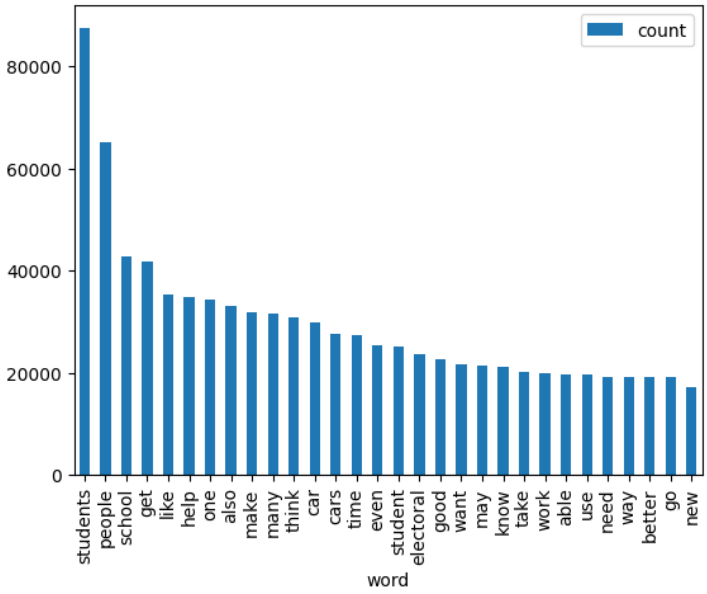


Figure 3. Common words of human written essays

Lastly, we will analyze the common words in human written essays. After removing stop words in English, we count the appearance of each word. Then, we plot the words and their frequencies as in Figure 3. The top five most common words are “students”, “people”, “school”,

"get" and "like". The initial analysis allows us to have an overall understanding of the source text written by human.

3.2 LLMs generated Essays

The rest of the corpus is generated by various large language models. We used a total of six LLMs to produce the source texts. Namely, they are GPT 3.5, GPT 4, LLaMA, Falcon, Claude and PaLM. ChatGPT 3.5 and ChatGPT 4 are developed by OpanAI in the year 2022 and 2023 (ChatGPT-Release Notes, 2023). PaLM was announced last year by Google AI (Pathways language model, 2022). LLaMA is released by Meta AI in current year (Touvron, 2023). Falcon (Tiiuae, 2023) and Claude (Introducing Claude, 2023) are published this year by Technology Innovation Institute in Abu Dhabi and AI assistance company called Anthropic. The mixture of LLMs ensures the sophistication and comprehensiveness of the capability of language models. Its diversification makes sure that our dataset includes all kinds of language nuances, thematic variations and linguistic structures. When generating the essays by LLMs, we offer the prompt name to the models. The output articles are collected into our dataset. To be more specific the prompt names are the same as the ones using in the human written essays' dataset.

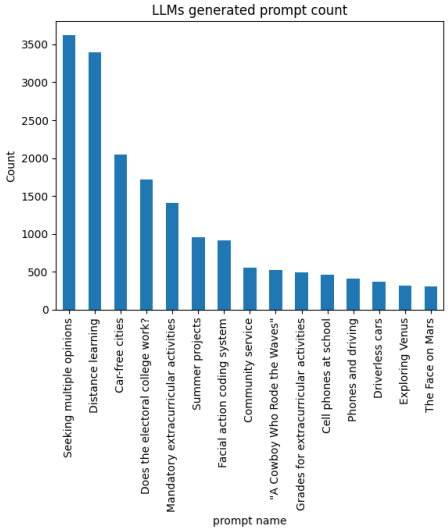


Figure 4. LLMs generated prompt count

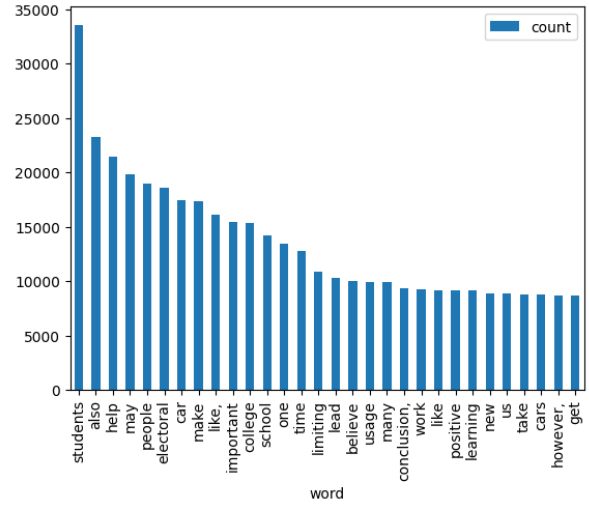
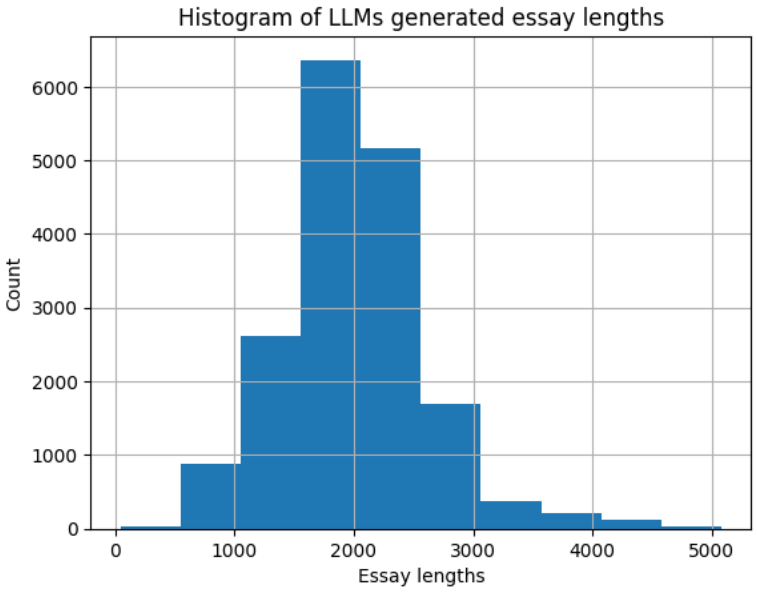


Figure 5. Common words of LLMs generated essays

We perform similar exploratory analysis on the LLMs generated essays. The figure 4 shows the top three prompts are "Seeking multiple opinions", "Distance learning" and "Car-free cities". In the Figure 6 and table 2, we can see the medium of the essay length is 1973 and the mean is

2009. The figure 5 shows the top five common words used by the LLMs are “students”, “also”, “help”, “may” and “people”.

Figure 6. Histogram of LLMs generated essay lengths



count	17497.000000
mean	2009.292450
std	603.496414
min	48.000000
25%	1650.000000
50%	1973.000000
75%	2350.000000
max	5078.000000

Table 2. Descriptive statistics of human written essay lengths

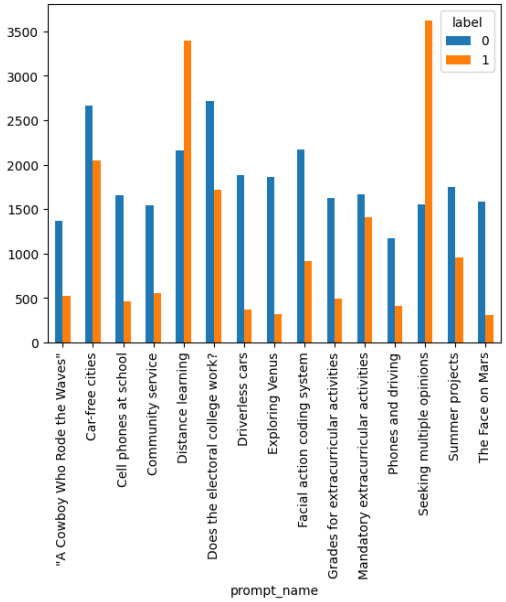
3.3 Comparison between the essays produced by human and LLMs

label	word_count	unique_word_count	sentence_count	avg_word_length
0	418.283146	199.858646	21.651748	4.531510
1	329.398983	169.369092	18.198891	5.093551

Table 3. Comparison on Unique words

To better learn the key difference between the human written essays and generated ones. We analyze several main aspects the essays. First, we count the essays by different producers. The figure 7 shows their distribution on the various prompt, which later helps us to explore human and LLMs different patterns on specific topics. Next, the figure 8 shows that human and LLMs created similar length essays, while human’s essay lengths have a wider spread and more outliers. In table x, human used more unique words than LLMs do. The figure 3 and the figure 5 which introduced earlier shows human and LLMs have a distinct taste of word chosen, which can be an inspiration in later research. The side-by-side analysis help us reveal unique aspects of

LLMs generated and human written texts. These comparison and contrast help highlight the characteristics and trends which differentiate human and LLMs.



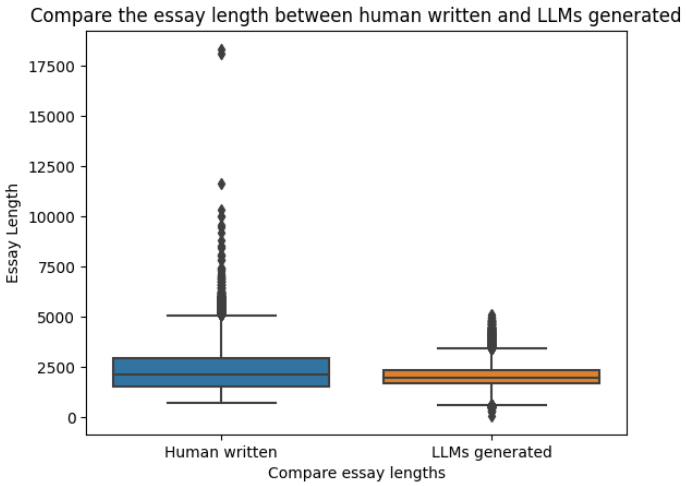


Figure 7. Comparison on prompt names Figure 8. Comparison on essay lengths

3. Methods

4.1 Data Preprocessing

We first cleaned the dataset by removing all characters that cannot be generated by humans. We sampled from both human generated text and AI generated text and found the difference between the two sets. We noticed this character includes Unicode that can be rendered as emoji and newline character. Then, we removed all the characters that cannot be generated by humans from the AI generated text dataset. The removal is done to ensure that our detectors do not rely on superficial signals like those special characters and look for deeper and potentially more accurate signals from the text. Without such preprocessing, it is easy to imagine an adversarial algorithm that can evade such detectors by carefully avoiding impossible characters.

After that, we performed normalization with lowercase and Unicode Normalization Form Canonical Composition (NFC) normalizer. The lowercase normalizer makes sure all inputs are transformed into lowercase characters while the NFC normalizer ensures the variants of Unicode characters are remapped by canonical equivalence. Both normalizers are common in NLP related tasks to make sure our detector can generalize well and is invariant to the superficial signals like lower and upper case.

Similarly, it is also worth correcting any typos from both the human and AI generated dataset to prevent detectors overfitting on superficial signals like typos. Since human generated text contain more typos by nature, a naive detector that just looks for the number of typos can perform well. However, it is also easy to imagine an adversarial AI algorithm to carefully include typos with some probability to mimic the mistakes that human can make in writings.

Although there are libraries that can correct typos, due to licensing restrictions and time constraint, we decided to not use them. Instead, we chose to use a carefully designed tokenizer that takes typo into consideration. Details on the custom tokenizer will be discussed later.

4.2 Out-of-distribution Learning

Initially, we used standard stratified random split on the dataset to get train and validation sets, but we noticed the area under the receiver operating characteristic curve (AUC-ROC) being very high (> 0.99), even for simple regression models like logistic regression, shown in Figure 8. We postulated that the in-distribution learning may be too easy since we train and valid with the essays for the same set of prompts. For a given prompt, certain characteristic word or phrases may be more prevalent, which gives our model a good signal in the embedding space to differentiate the human V.S. AI generated text. As a result, we decided to reserve one prompt just for validation too see how well our models behave in a more realistic out-of-distribution learning setting. As (Antoun, 2023), (Li, 2023) point out, even for sophisticated detectors like DetectGPT (Mitchell, 2023) and GLTR (Gehrmann, 2019) shows significant performance decrease when faced with out-of-distribution data, in Figure 9.

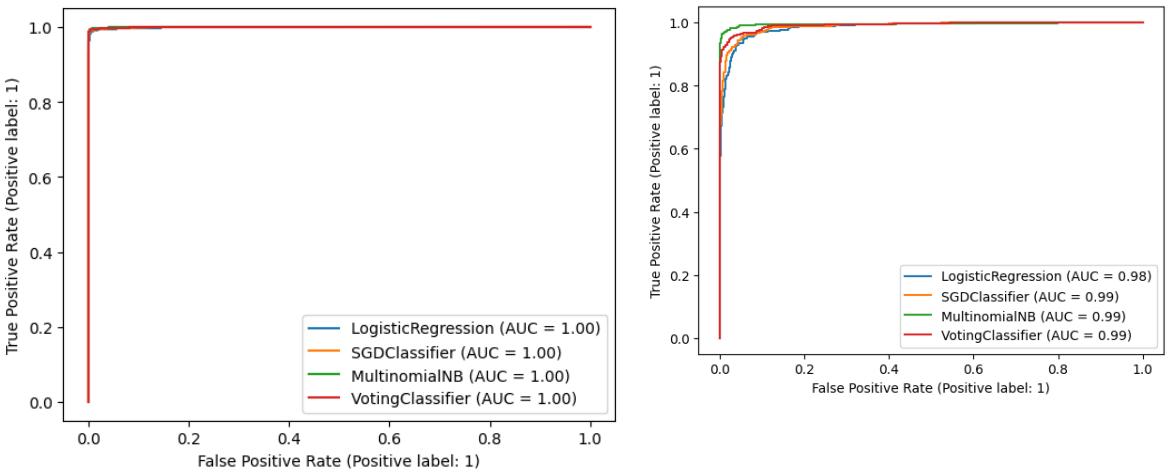


Figure 9. In-distribution-roc plot Figure 10. Out-distribution-roc plot

4.3 Feature Extraction

4.3.1 TF-IDF

We first explored the classical method: term frequency-inverse document frequency (TF-IDF) n-grams as the vectorizer to extract feature. TF-IDF is a common way to measure the importance of a word to a document in a corpus.

We also explored the n-gram range that controls the number of words that consist of the phrase and used different number of words and get their TF-IDF value respectively, since rather than a single word, phrases sometimes may better reflect a person (or a LLM)'s writing style.

4.3.2 Custom Tokenizer

As mentioned above, due to the lack of publicly available libraries to detect and correct typos, we also explored a custom version of tokenizer that takes into account of typos. Although classical TF-IDF vectorization performs well normally, when the text contains typos, the vocabulary size will greatly increase, making machine learning models less effective. For example, consider these two words: "university" (correct) and "universaty" (typo). If we were to apply the TF-IDF vectorization, we may undercount the score for "university" while introducing unwanted noise by assigning a non-zero score to "universaty". As (Sennrich, 2016) discover, byte-pair encoding (BPE) is an effective segmentation technique to address the out-of-vocabulary word problem by "encoding rare and unknown words as sequences of subword units". Since the issue with typo that we are facing can be framed as the rare word in the out-of-vocabulary word problem, we may take advantage of BPE to encode typos as sequences of subword units.

4. MODEL

5.1 Logistic Regression

Logistic Regression models the probability of an event happening by the log-probability of the event as a linear combination of multiple variables. We used the LogisticRegression model provided by scikit-learn (Pedregosa, 2011) and used "liblinear" as the solver in the base model since we have a relatively small dataset. Also, "liblinear" solver supports both L1 and L2 regularization, which gives us more control in the hyperparameter tuning.

5.2 Stochastic Gradient Decent Training on Linear Models

Stochastic Gradient Decent is a very fast iterative method that optimize for a loss function. We used the SGDClassifier model provided by scikit-learn (Pedregosa, 2011) and used "log_loss" as the loss in the base model since it is a popular choice for binary classification problems.

5.3 Multinomial Naive Bayes

Although Naive Bayes model assumes the predictors to be conditionally independent and all features contribute equally to the outcome, it is proven to be useful in real-world classification problems. is a We used MultinominalNB model provided by scikit-learn (Pedregosa, 2011) and used 0.02 as the base alpha smoothing parameter. Naive Bayes classifier works for both integer feature counts and fractional counts so the vectorized TF-IDF feature will also work.

5.4 Gradient Boosting

Gradient Boosting is a decision tree based method which combines multiple trees together in a additive training process. We used XGBoost model provided by DMLC XGBoost (Chen, 2016) and used "binary:logistic" as the objective since we are doing binary classification. We chose 100 as the base "n_estimator" parameter.

5.5 Transformer

We used "deberta-v3-xsmall" provided by Microsoft (He, 2021), which "improves BERT and RoBERTa models using disentangled attention and enhanced mask decoder.". We chose the extra small model to be efficient. The model has 12 layers and only 22M backbone parameters. Despite its small size, it has been shown that its performance is on par with even the much larger RpBERTa-base model.

5.1 Ensemble Model

The idea behind an ensemble model is simple. We take advantage of the output from multiple models and combine the output probability score with a weighted sum. We used the VotingClassifier model provided by scikit-learn (Pedregosa, 2011) and used "soft" as the base voting strategy.

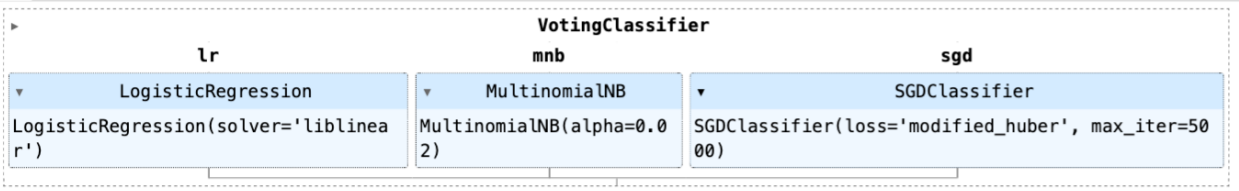


Figure 11. Ensemble architecture

5. RESULTS

The table 3 below offers a summary on the results for the different models and techniques explored. Since we are treating this problem as a binary-classification problem, we use the AUC-ROC as the metric to evaluate the performance of our model. The higher the AUC means the better the performance that the model has. As mentioned above, we purposely reserved an out-of-distribution test dataset to make this problem more challenging and closer to real world applications.

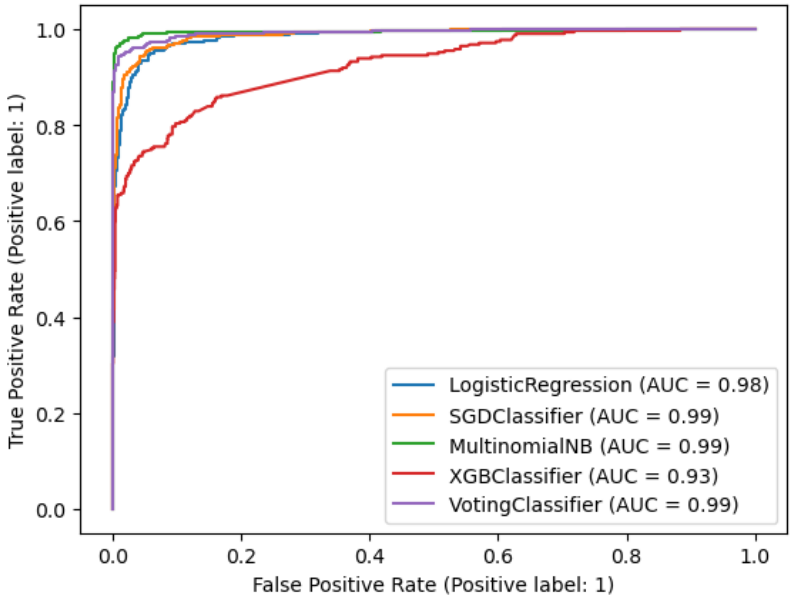


Figure 12. Summary of the results

6.1 Custom Tokenizer

We innovatively used a custom version of tokenizer that significantly boosts the AUC on the out-of-distribution dataset. The reason behind such good performance is that the customized BPE tokenizer help the model to generalize well by encoding typos, which are superficial characteristics from human writing, as sequences of subword units. We used Tokenizers library by Huggingface [38]. We chose 30522 as the vocabulary size to match the value used in BERT (Chang, 2018). Below are plots showing the significance of such a custom tokenizer. All models, except for Multinomial Naive Bayes and Gradient Boosting models receive a decent performance boost.

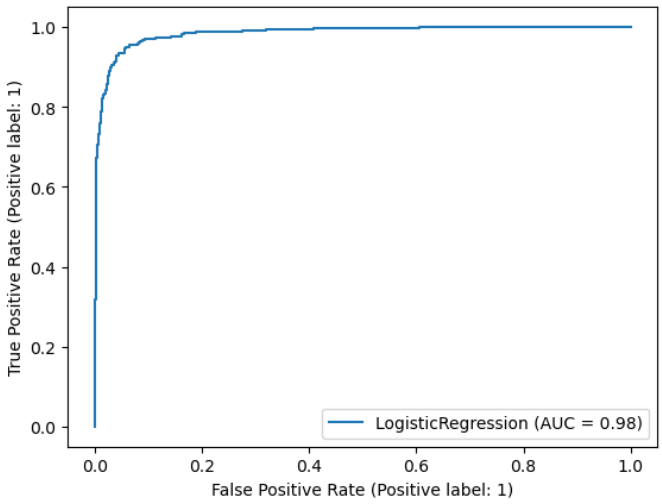


Figure 13. Logistic Regression - Base

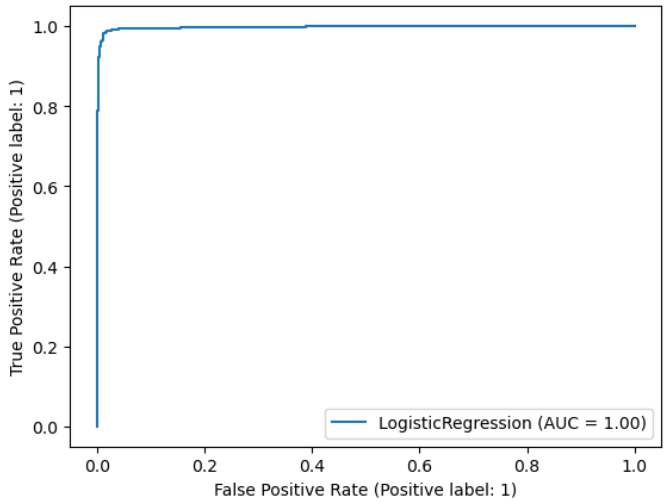


Figure 14. Logistic Regression with tokenizer

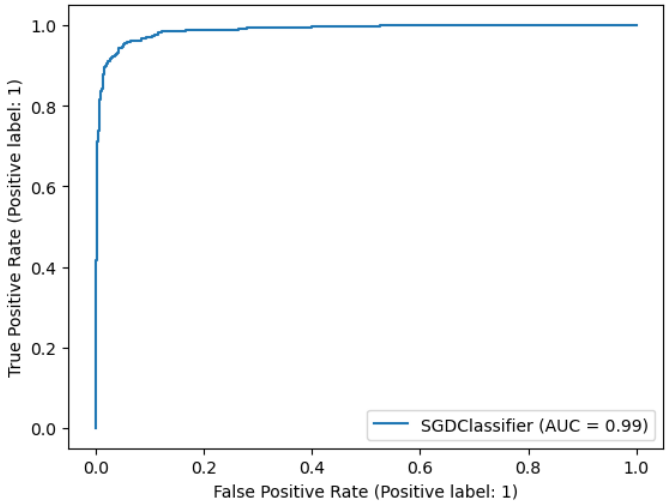


Figure 15. SGD Classifier - Base

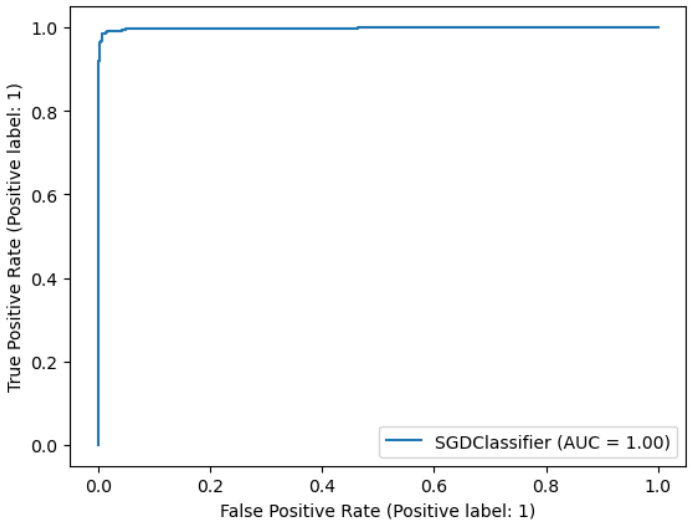


Figure 16. SGD Classifier with tokenizer

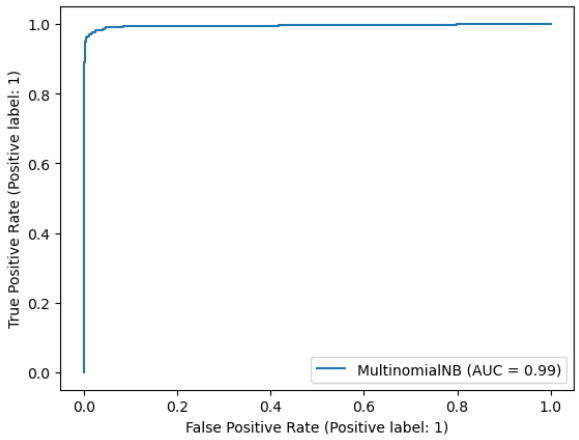


Figure 17. MultinomialNM - Base

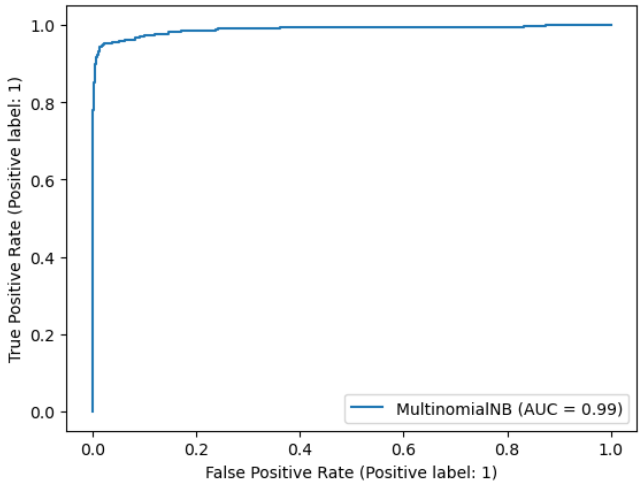


Figure 18. MultinomialNM with tokenizer

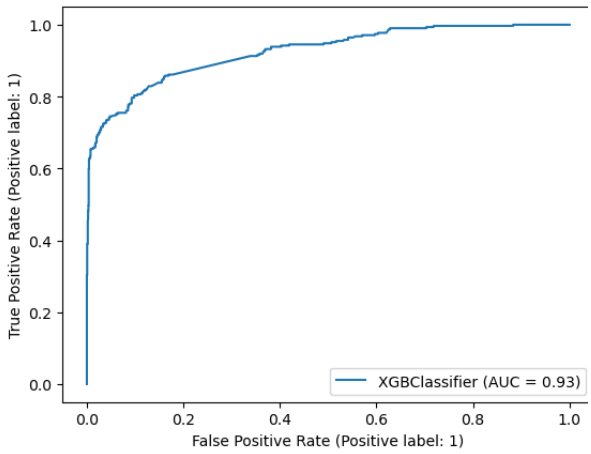


Figure 19. XGBClassifier - Base

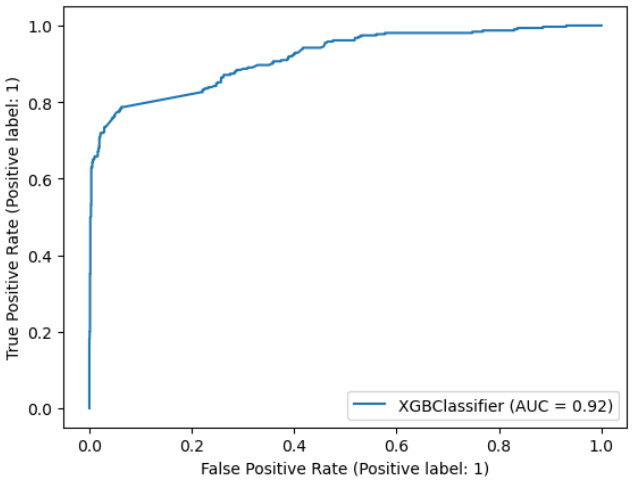


Figure 20. XGBClassifier with tokenizer

We also explored a variety of hyper-parameters for each model. The sections below will elaborate on the experiments we performed with each model.

6.2 Logistic Regression

We chose "liblinear" solver which gives us access to both l1 and l2 regularization. We explored the different choices of regularization: l1 only, l2 only and both l1 and l2. Note for the combined l1 and l2 penalty, we used "saga" solver and chose the l1 penalty ratio to be 0.5.

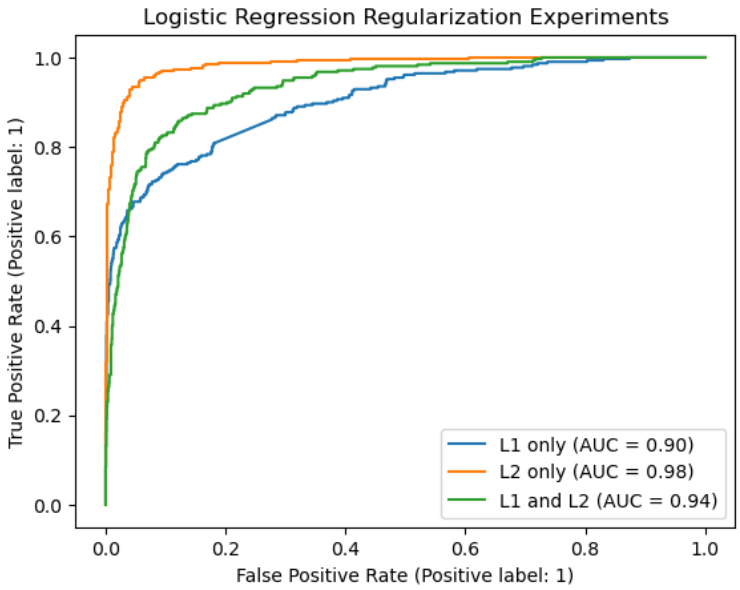


Figure 21. Logistic Regression Regularization Experiments

6.3 Stochastic Gradient Decent

There are multiple choices of loss functions that are suitable for classification problems. We explored the different loss functions. We found out Squared Hinge and Modified Huber are two of the best performing loss functions.

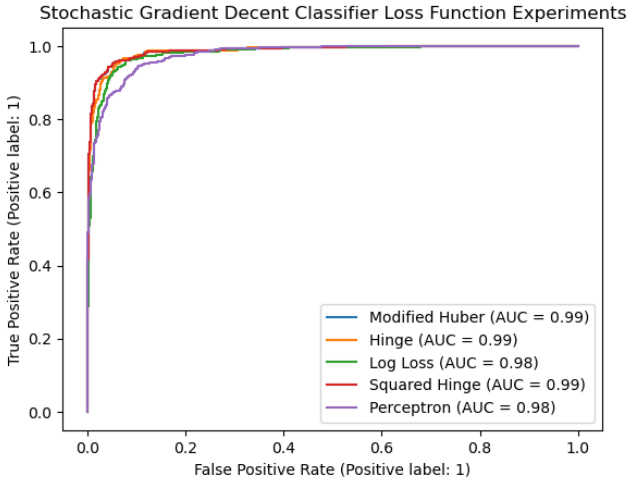


Figure 22. Stochastic Gradient Decent Classifier Loss Function Experiments

Based on the best performing loss function Squared Hinge, we also explored the effect of penalty on the model.

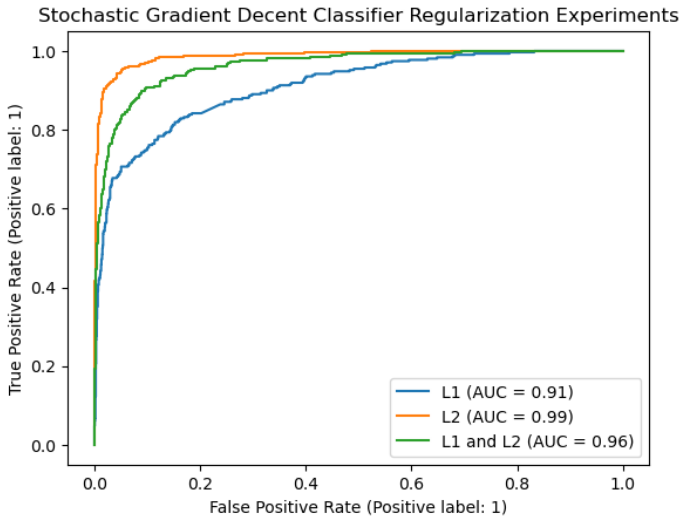


Figure 23. Stochastic Gradient Decent Classifier Loss Function Experiments with penalty

6.4 Multinomial Naïve Bayes

We explored the "alpha", which is additive smoothing parameter. We probed values from 0.01 to 0.04 with 0.01 increment. We found that alpha=0.1 gives us the best result.

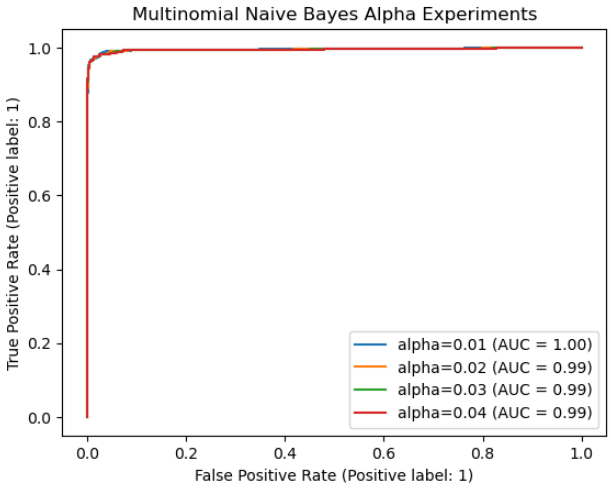


Figure 24. Multinomial Naïve Bayes Alpha Experiments

6.5 Gradient Boosting

There are a wide range of hyperparameters available to tune, including "n_estimators", which represents the number of boosting rounds, "max_depth", which represents the tree depth for base learners etc. Due to time and resource constraint, we did not perform extensive hyperparameter experimentation since each round of training will take about 30 mins on CPU. Below is the best

XGBoost model we observed with an AUC of 0.93. Granted, there may be abudent areas for improvement for the XGBoost model and we plan to further investigate this idea in the future.

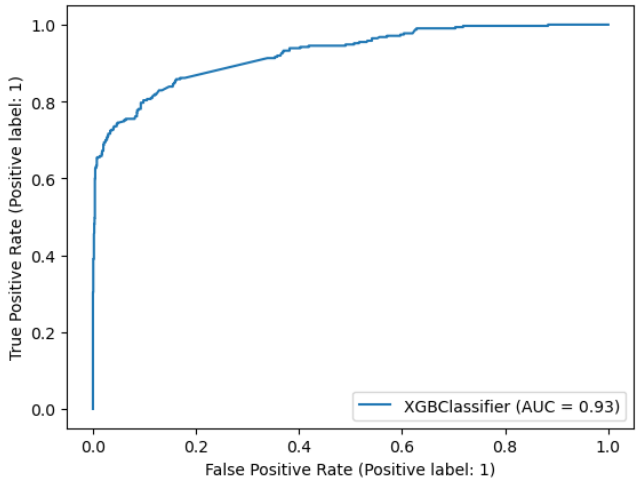


Figure 25. XGB ROC

6.6 Transformer

Thanks to the recent LLM advance, research and new models on Transformers are emerging at a very fast pace, leading to no shortage of ideas, models and hyperparameters to experiment with. Some of the most common hyperparameters includes learning rate, batch size, gradient accumulation steps, weight decay etc. Here we chose a moderate hyperparameter to ensure efficient training while getting a decent result. We set train batch size as 4 and gradient accumulation steps as 32. Undoubtedly, given the success of transformer models on a wide range of real-world tasks, Transformer models have great potentials. In practice in this problem, we noticed its ROC curve is one of the best performing one. With enough time and resource tuning and searching the hyperparameter space, we believe Transformers can be very accurate and reliable in this task.

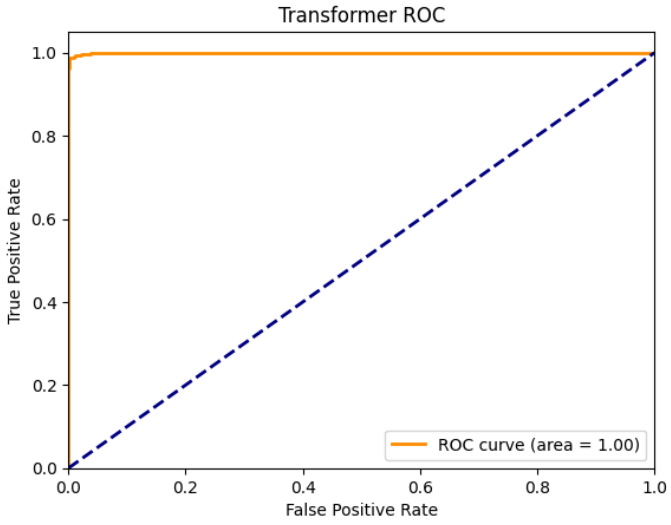


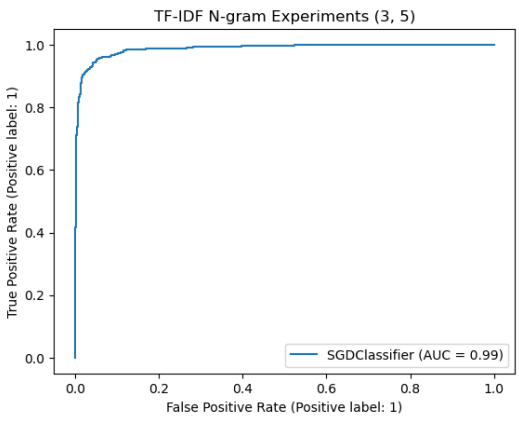
Figure 26. Transformer ROC

6.7 Ensemble Model

Due to time and resource constraint, we focused our experiments with ensemble model on the simple and efficient models like Logistic Regression, Stochastic Gradient Decent and Multinomial Naive Bayes models. As Figure 12 shows, we found that the ensemble model slightly behaved worse than the best performing Multinomial Naive Bayes model. It may be due to the poorly optimized XGBoost model influencing the final decision. We explored both soft and hard voting strategy and found the soft voting strategy to be better. As we optimize the individual sub-models, the gap between soft and hard voting should be larger since soft strategy usually performs better on well-calibrated classifiers (Pedregosa, 2011).

6.8 TF-IDF N-Gram

We also explored the effect of N-gram in the TF-IDF feature extraction process. The N-gram controls the length of the phrases we use to vectorize the text. For example, if N=1 then we are extracting words and if N=2 we are extracting phrases that consist of 2 words. Thanks to the flexible "TfidfVectorizer" that scikit-learn provides, we can provide a tuple to represent the lower and upper boundary of the range of n-values for different n-grams to be extracted (Pedregosa, 2011). We chose Stochastic Gradient Decent models as the benchmarking model while exploring a few options for n-value and noticed significant impact on the AUC score. This phenomenon implies the significance of characteristic phrases which often reflects one's unique writing style. In the future, we decide to further optimize and find a better n-value. Perhaps with a more complex and capable model like transformers, we may be able to take advantage of a large n-value which encodes richer and hidden information as the word being farther away.



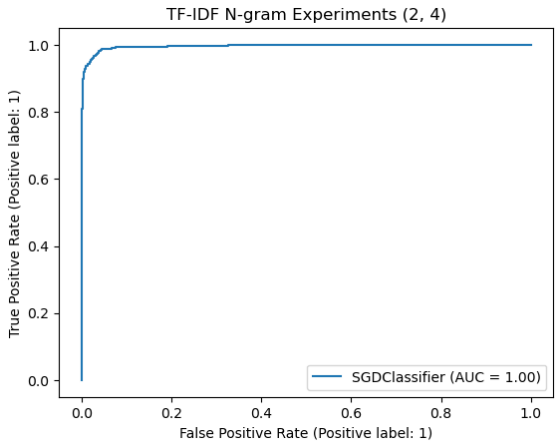


Figure 27. TF-IDF N-Gram Experiments (3,5) Figure 28. TF-IDF N-Gram Experiments (2,4)

6. DISCUSSION & CONCLUSION

In conclusion, we explored a wide range of models, from simple linear models like Logistic Regression and Stochastic Gradient Decent to more complex models such as DeBERTa transformers. We also probed hyperparameter choices for each model. More importantly, we developed a customized BPE based tokenizer that is proven to be very effective at addressing the large vocabulary size introduced by typos.

Due to time and resource constraint, we left many interesting ideas unexplored. One of the most promising ideas is to take advantage of Transformers, which have been proven to be very effective at NLP tasks. Even with the limited training, our transformers are among the top performers. Future study can explore the use of larger Transformer models in the DeBERTa model family. Also, as shown by (Glazkova, 2023), an ensemble model with DeBERTa, SciBERT and RoBERTa has proven to be effective in identifying machine generated scientific papers, so we can explore transformer-based ensemble models. Since the model complexity is significantly higher than other simpler models, we can also explore potential ways of augmenting data by probing more choices of N-gram (Lee, 2023).

On the other hand, our custom tokenizer which already is proven to be quite effective on many models also has plenty room for improvement. As (Bostrom, 2020) points out, BPE's greedy construction procedure is suboptimal compared to unigram language model tokenizer. We hope to explore the unigram tokenizer and their effect on our models. More interestingly, we plan to explore Transformers that are pre-trained with unigram tokenizer (Zevallos, 2023) and compare the difference in performance with respect to the Transformers since it was shown the transformers that are pre-trained with unigram tokenizer have superior results in a wide range of downstream tasks.

REFERENCES

[1] OpenAI (2023). GPT-4 Technical Report. https://cdn.openai.com/papers/gpt-4.pdf

[2] Scanlon, M., Breitinger, F., Hargreaves, C., Hilgert, J.-N., & Sheppard, J. (2023). ChatGPT for digital forensic investigation: The good, the bad, and the unknown. Forensic Science International: Digital Investigation, 46, 301609. https://doi.org/10.1016/j.fsidi.2023.301609

[3] Crossley, S., Baffour, P., Yu, T., Franklin, A., Benner, M., & Boser, U. (2023). A large-scale corpus for assessing written argumentation: PERSUADE 2.0. https://doi.org/10.1016/j.asw.2023.100667

[4] Darek Kleczek. (2023). DAIGT V2 Train Dataset, Version 2. Retrieved train_v2_drcat_02.csv from https://www.kaggle.com/datasets/thedrcat/daigt-v2-train-dataset/data

[5] ChatGPT-Release Notes from the original on October 2, 2023. https://help.openai.com/en/articles/6825453-chatgpt-release-notes#h_4799933861

[6] Pathways language model (Palm): Scaling to 540 billion parameters for breakthrough performance. (2022, April 4). https://blog.research.google/2022/04/pathways-language-model-palm-scaling-to.html

[7] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., & Lample, G. (2023). Llama: Open and efficient foundation language models (arXiv:2302.13971). arXiv. https://doi.org/10.48550/arXiv.2302.13971

[8] Tiiuae/falcon-refinedweb · datasets at hugging face. (2023, September 29). https://huggingface.co/datasets/tiiuae/falcon-refinedweb

[9] Introducing Claude. Anthropic. Mar 14, 2023. https://www.anthropic.com/index/introducing-claude

[10] Sennrich, R., Haddow, B., & Birch, A. (2016). Neural machine translation of rare words with subword units (arXiv:1508.07909). arXiv. https://doi.org/10.48550/arXiv.1508.07909

[11] Quicktour. (n.d.). Hugging Face. Retrieved December, 2023, from https://huggingface.co/docs/tokenizers/quicktour

[12] Antoun, W., Mouilleron, V., Sagot, B., & Seddah, D. (2023). Towards a robust detection of language model generated text: Is chatgpt that easy to detect? https://doi.org/10.48550/ARXIV.2306.05871

[13] Li, Y., Li, Q., Cui, L., Bi, W., Wang, L., Yang, L., Shi, S., & Zhang, Y. (2023). Deepfake text detection in the wild. https://doi.org/10.48550/ARXIV.2305.13242

[14] E. Mitchell, Y. Lee, A. Khazatsky, C. D. Manning, and C. Finn, "Detectgpt: Zero-shot machine-generated text detection using probability curvature," in International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA, ser. Proceedings of Machine Learning Research, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., vol. 202. PMLR, 2023, pp. 24 950–24 962. https://proceedings.mlr.press/v202/mitchell23a.html

[15] Gehrmann, S., Strobelt, H., & Rush, A. (2019). Gltr: Statistical detection and visualization of generated text. In M. R. Costa-jussà & E. Alfonseca (Eds.), Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations (pp. 111–116). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-3019

[16] He, P., Gao, J., & Chen, W. (2021). Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing.

[17] Glazkova, A., & Glazkov, M. (2022). Detecting generated scientific papers using an ensemble of transformer models (arXiv:2209.08283). arXiv. https://doi.org/10.48550/arXiv.2209.08283

[18] Bostrom, K., & Durrett, G. (2020). Byte pair encoding is suboptimal for language model pretraining (arXiv:2004.03720). arXiv. https://doi.org/10.48550/arXiv.2004.03720

[19] Lee, R. S. T. (2023). N-gram language model. In R. S. T. Lee (Ed.), Natural Language Processing: A Textbook with Python Implementation (pp. 19–42). Springer Nature. https://doi.org/10.1007/978-981-99-1999-4_2

[20] Zevallos, R., & Bel, N. (2023). Hints on the data for language modeling of synthetic languages with transformers. In A. Rogers, J. Boyd-Graber, & N. Okazaki (Eds.), Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 12508–12522). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.acl-long.699

[21] Zheng, Q., Xia, X., Zou, X., Dong, Y., Wang, S., Xue, Y., Wang, Z., Shen, L., Wang, A., Li, Y., Su, T., Yang, Z., & Tang, J. (2023). Codegeex: A pre-trained model for code generation with multilingual evaluations on humaneval-x (arXiv:2303.17568). arXiv. https://doi.org/10.48550/arXiv.2303.17568

[22] Introducing GitHub Copilot: your AI pair programmer. The Github Blog. June 2021. https://github.blog/2021-06-29-introducing-github-copilot-ai-pair-programmer/

[23] Murakami, S., Hoshino, S., & Zhang, P. (2023). Natural language generation for advertising: A survey (arXiv:2306.12719). arXiv. https://doi.org/10.48550/arXiv.2306.12719

[24] Susnjak, T. (2022). Chatgpt: The end of online exam integrity? (arXiv:2212.09292). arXiv. https://doi.org/10.48550/arXiv.2212.09292

[25] Educators Battle Plagiarism As 89% Of Students Admit To Using OpenAI's ChatGPT For Homework. Chris Westfall. Forbes. Jan 28, 2023. https://www.forbes.com/sites/chriswestfall/2023/01/28/educators-battle-plagiarism-as-89-of-students-admit-to-using-open-ais-chatgpt-for-homework/?sh=416a8534750d

[26] Pagnoni, A., Graciarena, M., & Tsvetkov, Y. (2022). Threat scenarios and best practices to detect neural fake news. In N. Calzolari, C.-R. Huang, H. Kim, J. Pustejovsky, L. Wanner, K.-S. Choi, P.-M. Ryu, H.-H. Chen, L. Donatelli, H. Ji, S. Kurohashi, P. Paggio, N. Xue, S. Kim, Y. Hahm, Z. He, T. K. Lee, E. Santus, F. Bond, & S.-H. Na (Eds.), Proceedings of the 29th International Conference on Computational Linguistics (pp. 1233–1249). International Committee on Computational Linguistics. https://aclanthology.org/2022.coling-1.106

[27] Lin, S., Hilton, J., & Evans, O. (2022). Truthfulqa: Measuring how models mimic human falsehoods. In S. Muresan, P. Nakov, & A. Villavicencio (Eds.), Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 3214–3252). Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-long.229

[28] Weidinger, L., Mellor, J., Rauh, M., Griffin, C., Uesato, J., Huang, P.-S., Cheng, M., Glaese, M., Balle, B., Kasirzadeh, A., Kenton, Z., Brown, S., Hawkins, W., Stepleton, T., Biles, C., Birhane, A., Haas, J., Rimell, L., Hendricks, L. A., … Gabriel, I. (2021). Ethical and social risks of harm from Language Models (arXiv:2112.04359). arXiv. https://doi.org/10.48550/arXiv.2112.04359

[29] Ayoobi, N., Shahriar, S., & Mukherjee, A. (2023). The looming threat of fake and llm-generated linkedin profiles: Challenges and opportunities for detection and prevention. Proceedings of the 34th ACM Conference on Hypertext and Social Media, 1–10. https://doi.org/10.1145/3603163.3609064

[30] Mirsky, Y., Demontis, A., Kotak, J., Shankar, R., Gelei, D., Yang, L., Zhang, X., Pintor, M., Lee, W., Elovici, Y., & Biggio, B. (2023). The threat of offensive ai to organizations. Computers & Security, 124, 103006. https://doi.org/10.1016/j.cose.2022.103006

[31] Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., & Fung, P. (2023). Survey of hallucination in natural language generation. ACM Computing Surveys, 55(12), 248:1-248:38. https://doi.org/10.1145/3571730

[32] Lee, J., Le, T., Chen, J., & Lee, D. (2023). Do language models plagiarize? Proceedings of the ACM Web Conference 2023, 3637–3647. https://doi.org/10.1145/3543507.3583199

[33] Sadasivan, V. S., Kumar, A., Balasubramanian, S., Wang, W., & Feizi, S. (2023). Can ai-generated text be reliably detected? (arXiv:2303.11156). arXiv. https://doi.org/10.48550/arXiv.2303.11156

[34] Mitchell, E., Lee, Y., Khazatsky, A., Manning, C. D., & Finn, C. (2023). Detectgpt: Zero-shot machine-generated text detection using probability curvature. Proceedings of the 40th International Conference on Machine Learning, 24950–24962. https://proceedings.mlr.press/v202/mitchell23a.html

[35] Gehrmann, S., Strobelt, H., & Rush, A. (2019). Gltr: Statistical detection and visualization of generated text. In M. R. Costa-jussà & E. Alfonseca (Eds.), Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations (pp. 111–116). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-3019

[36] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in python. Journal of Machine Learning Research, 12(85), 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html

[37] Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794. https://doi.org/10.1145/2939672.2939785

[38] He, P., Liu, X., Gao, J., & Chen, W. (2021). Deberta: Decoding-enhanced bert with disentangled attention (arXiv:2006.03654). arXiv. https://doi.org/10.48550/arXiv.2006.03654

[39] Tokenizer. (n.d.). Retrieved December 4, 2023, from https://huggingface.co/docs/transformers/main_classes/tokenizer

[40] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding (arXiv:1810.04805). arXiv. https://doi.org/10.48550/arXiv.1810.04805