AI-Generated Text Classification in Scientific Articles

Drew Christner (dzchrist@asu.edu)
November 24, 2023
CSE 472 - Social Media Mining - Project 2 Report

1. Abstract

This report introduces a machine learning model intended to identify and classify AI generated scientific misinformation within the text of scholarly articles. The primary task is the classification of the text as either created by humans or artificial intelligence. The classification is done using the text from the title, abstract, and introduction sections of scientific papers which are either human written or AI generated. The growing concerns surrounding the use of large language models(LLM) for the spread of scientific misinformation highlights the importance of developing tools for the detection of AI generated content.

In order to develop this model, the exploration of a variety of feature extraction techniques which were tested and compared to select the most useful ones. The methods explored include TF-IDF, count vectorization, and n-gram representations. The accuracy score and F1 metric of various classifications of these features were compared and the highest scoring utilized in the final model.

Development also involved experimentation with a variety of different classification algorithms/methods to identify the one that yielded the best accuracy and F1 score. Classification methods including Naive Bayes, Logistic Regression, and Support Vector Machines were explored and compared to identify the best model architecture.

After exploration and experimentation with these feature extraction and classification methods, the design for the final model was created. This model utilizes a Voting Classifier to combine Support Vector Machine and Logistic Regression classifiers which will classify the TF-IDF vectors of the introduction portions of the input data. The experimental results reveal the efficacy of the proposed model in accurately classifying AI-generated scientific misinformation. The selected feature extraction techniques, in conjunction with the classification algorithm, yielded an F1 score of over 92%. This research contributes to the ongoing efforts to address the challenges posed by AI-generated misinformation in scientific literature.

2. Introduction

In the era of rapidly advancing technology, the use of artificial intelligence for the generation of scientific discourse has become a focal point of both innovation and concern. This report addresses a facet of this intersection by introducing a machine learning model designed to discern and categorize AI-generated scientific misinformation within the text of scholarly articles. The primary objective of this model is the classification of text, distinguishing between articles created by human authors and those generated by artificial intelligence.

Large language models have fostered concerns over the potential for the dissemination of misinformation within scholarly contexts, further fueling widespread concerns about the power and influence of AI. Recognizing the need for vigilant safeguards in the scientific discourse landscape, this study aims to contribute a tool for the detection and classification of AI-generated content. Specifically, this classifier is directed towards the text within the titles, abstracts, and introductions of scientific papers.

To develop this model, a comprehensive exploration of feature extraction techniques was undertaken. Various methodologies, including Term Frequency-Inverse Document Frequency (TF-IDF), count vectorization, and n-gram representations, were systematically evaluated. The selection of the most informative features was based on a thorough comparative analysis of accuracy scores and F1 metrics across diverse classifications.

Beyond feature extraction, the model's architecture was refined through experimentation with a variety of classification algorithms and methods. This entailed the exploration of techniques such as Naive Bayes, Logistic Regression, and Support Vector Machines, with a meticulous comparison to identify the optimal model configuration. The pursuit of the highest accuracy and F1 score underpins the methodological choices, ensuring a robust and effective tool for the identification of AI-generated scientific misinformation.

In an age where the boundaries between human and machine-generated content continue to blur, the development of tools to safeguard the integrity of scientific discourse becomes imperative. This report displays the journey of constructing a machine learning model that not only addresses this imperative but also contributes to the ongoing discourse on the responsible use of AI in scientific communication. Through a combination of feature extraction techniques and model selection, this study strives to provide a reliable means for researchers, academics, and the

broader scientific community to navigate the evolving landscape of information dissemination.

3. Related Works

The development of LLMs has also led to accelerated efforts to produce tools that are able to automatically detect AI-generated text. Firstly, there have been several studies performed that look into the ability of humans to differentiate between AI and human created texts. For example, a 2023 study by Biyang Guo et al. investigated multiple essential questions which compared ChatGPT to humans including: what are the characteristics of ChatGPT's responses, what similarities exist between human experts and ChatGPT, how do the linguistics of ChatGPT compare to humans, and lastly how can AI-generated text be effectively detected.

There are also many recent articles that focus on studying and improving AI-generated text detection methods. A 2023 research article by Debora Weber-Wulff et al. investigated 12 publicly available AI-generated text detection tools as well as two large commercial systems, Turnitin and PlagiarismCheck. After extensive testing of these tools, the researchers found that they are largely inaccurate and unreliable, displaying a significant bias towards classifying inputs as human written. Furthermore, the researchers also investigated content obfuscation techniques, such as manual edits to text after AI-generation, and how this affects the accuracy of the detection tools. This investigation showed that any content obfuscation further reduced the accuracy and reliability of detection.

These previous studies highlight the difficulty that the task of differentiating AI and human created content presents. They also reveal the inadequacy of the currently available tools that are used to automate the detection of AI-generated content. These works have shown a high degree of inaccuracy and a lack of reliability within these tools as most of them contain a bias towards marking text as human, even if it was generated by an AI. The model presented in this report is intended to present an additional tool which can help

4. Model Description

After the experimentation process described in the following section, the final text classification model was developed. This model takes raw text data from the title, introduction, and abstract sections of scientific articles as inputs. It then uses this data to classify the inputs as either created by

humans(0) or generated by AI(1). To compute the classification predictions, the raw text must be converted into a feature vector that can be fed into classification algorithms. The feature selected for use in this model is the TF-IDF of just the introduction portion of the input. Although the title, abstract, introduction, and the combination of the three were tested for use as TF-IDF feature vectors, using the introduction section alone produced the best results. To classify this extracted feature vector, an ensemble learning system was implemented. Specifically, a voting classifier that utilizes two different classification algorithms to produce the final prediction is implemented in the final model. The voting classifier runs the feature vector through both a C-Support Vector Classifier(SVC) and Logistic Regression and then performs soft, weighted voting on the results of the two classifiers. The SVC has a weight of 1.0 and the Logistic Regression has a weight of 0.9. After being trained on the training dataset and then applied to the test dataset, this model has an F1 score of just over 92% and an accuracy score of about 93%.

5. Experiment

5.1 Preprocessing

The first step in preprocessing the training and testing datasets is to import them into the program as Pandas dataframes. This is a clean, concise way to store data within Python so that it is simple to utilize it in machine learning projects. Next, in order to experiment with the data, it needs to be split into a training and validation set. For the final model, the training data will not be split to provide the most data towards training as possible. To split the data, the function train_test_split from the sklearn library is used.

Next, the labels from the dataset are encoded so that they are normalized for use in later classification. Finally, the raw text data from the title, abstract, and introduction sections are converted to lowercase. This step is built into the feature extraction methods from sklearn. Lowercasing in this manner allows for the feature vectors to have less keys (e.g. one for “Cat” and one for “cat”) and thus produce better results with less resource consumption.

5.2 Feature Engineering

Feature engineering converts the raw text data into feature vectors which can be used by the classifiers to make predictions. In order to determine which features to use for the final model, a variety of different features were experimented with to select the ones that produce the most

accurate predictions. In this section, each of the features tested will be described and the results of experimentation reported.

The first feature that was tested is count vectors. These vectors represent the frequency of each term in each document in matrix form. The other features are multiple versions of Term Frequency Inverse Document Frequency(TF-IDF). This feature contains the relative importance of each term in each document for the entire corpus. Multiple TF-IDF vectors were created each representing different levels to see which provided the most accurate results. These levels include word level and n-gram level. Word level looks at the frequency of a word in a document while n-gram uses a combination of n terms together, in this case combinations of 2 and 3 terms.

5.3 Classifiers

Once the feature vectors are constructed from the raw text data, they are then used to train a classifier to make predictions. The first classifier is a Naive Bayes model implemented using sklearn's MultinomialNB classifier. This classifier is based on Bayes theorem and it assumes that each feature is completely independent of the presence or absence of another feature, hence the name "naive" bayes.

The second classifier is a logistic regression linear classifier. This classifier measures the dependent variable and a set of independent variables by predicting probabilities using a logistic function. This is implemented using sklearn's LogisticRegression classifier.

The third classifier is a Support Vector Machine. This is a supervised machine learning algorithm that extracts the best possible line that divides two classes. This is implemented using sklearn's C-Support Vector Classifier(SVC).

In order to utilize multiple classification methods, a Voting Rule Classifier was implemented. This classifier runs inputs through a set of classifiers and compares all of the results. These results count as 'votes' for what the final prediction will be. For this model, the voting classifier runs both logistic regression and SVC with a weight of 1 for SVC and a weight of 0.9 for logistic regression.

5.4 Results

Count Vector								
	Title		**Abstract**		**Introduction**		**Combined**	
	F1	Accuracy	F1	Accuracy	F1	Accuracy	F1	Accuracy
Naive Bayes	0.628	0.606	0.721	0.703	0.826	0.835	0.796	0.797
Logistic Regression	0.608	0.602	0.734	0.725	0.916	0.916	0.903	0.901
SVC	0.621	0.59	0.726	0.71	0.896	0.89	0.892	0.888
Ensemble	0.567	0.573	0.739	0.732	0.911	0.91	0.903	0.902

Word Level TF-IDF								
	Title		**Abstract**		**Introduction**		**Combined**	
	F1	Accuracy	F1	Accuracy	F1	Accuracy	F1	Accuracy
Naive Bayes	0.613	0.603	0.687	0.684	0.536	0.669	0.557	0.667
Logistic Regression	0.611	0.593	0.717	0.713	0.919	0.917	0.876	0.875
SVC	0.617	0.594	0.721	0.717	0.931	0.93	0.887	0.888
Ensemble	0.605	0.602	0.752	0.751	0.937	0.931	0.905	0.908

N-Gram Level TF-IDF								
	Title		**Abstract**		**Introduction**		**Combined**	
	F1	Accuracy	F1	Accuracy	F1	Accuracy	F1	Accuracy
Naive Bayes	0.536	0.669	0.723	0.686	0.808	0.826	0.791	0.81

Logistic Regression	0.528	0.553	0.729	0.701	0.869	0.863	0.867	0.87
SVC	0.433	0.555	0.718	0.687	0.866	0.869	0.825	0.833
Ensemble	0.571	0.584	0.719	0.696	0.861	0.87	0.830	0.843

6. Future Works

This report succeeded in presenting a model that works to detect AI-generated text within the title, introduction, and abstract sections of scientific articles. Even with this success, there are still many issues that LLMs present that have yet to be tackled. This model and other models like it, are not infallible. There is still the possibility that AI-generated content can escape detection, especially if content obfuscation techniques are employed. This brings up the necessity for further research into new, more reliable methods of detecting AI text. These new classifiers could utilize emerging machine learning and AI concepts like deep neural networks. Furthermore, as AI and LLMs continue to become more powerful and similar to humans, it is going to become harder and harder to detect AI-generated text. As such, it is vital that tools to detect AI content continue to receive the same efforts towards improvement that is received by LLMs. Lastly, it is likely impossible to create a 100% effective AI content detector and as such, it is important to place safeguards against the harmful use of AI and LLMs earlier in the process of content generation.

7. References

Bansal, S. (2022, June 15). *A comprehensive guide to understand and implement text classification in Python*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2018/04/a-comprehensive-guide-to-understand-and-implement-text-classification-in-python/

Brownlee, J. (2021, May 7). *How to develop a weighted average ensemble with python*. MachineLearningMastery.com. https://machinelearningmastery.com/weighted-average-ensemble-with-python/

Guo B, Zhang X, Wang Z, Jiang M, Nie J, Ding Y, Yue J, Wu Y (2023) How Close is ChatGPT to Human Experts? Comparison Corpus, Evaluation, and Detection. arXiv. doi:10.48550/arXiv.2301.07597.

Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 0.1038/s41586-020-2649-2.

Heikkilä, M. (2022, December 19). *How to spot AI-generated text*. MIT Technology Review. https://www.technologyreview.com/2022/12/19/1065596/how-to-spot-ai-generated-text/

OpenAI. (2023). ChatGPT (Mar 14 version) [Large language model]. https://chat.openai.com/chat

Practical text classification with python and keras. Real Python. (2021, August 6). https://realpython.com/python-keras-text-classification/

Rogers, R. (2023, February 8). *How to detect AI-generated text, according to researchers*. Wired. https://www.wired.com/story/how-to-spot-generative-ai-text-chatgpt/

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

Srivastava, T. (2020, June 26). *Basics of Ensemble Learning explained in simple English*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2015/08/introduction-ensemble-learning/

Sudalai R. (2019, August 20). *Getting started with text preprocessing*. Kaggle. https://www.kaggle.com/code/sudalairajkumar/getting-started-with-text-preprocessing

Weber-Wulff, D., Anohina-Naumeca, A., Bjelobaba, S., Foltýnek, T., Guerrero-Dib, J., Popoola, O., Šigut, P., & Waddington, L. (2023, July 10). *Testing of detection tools for AI-generated text*. arXiv.org. https://arxiv.org/abs/2306.15666

8. Use of Generative AI in Report

To complete the report for this project, ChatGPT was used to edit and improve the written elements. This process involved writing a draft of each section of the report and then providing that as a prompt to ChatGPT and asking it to recommend changes and improvements to make to the writing. These changes were then revised again by the student and those final versions used in this report