ISyE 6420
Bayesian Statistics
Kai Kleinbard
November 28, 2023
Detecting AI Generated Text with Bayesian Analysis

Detecting AI Generated Text with Bayesian Analysis

Inspired by the Kaggle Competition: LLM Detect AI Generated Text

Table of Contents

Introduction

Bayesian models can be a powerful tool for distinguishing between AI-generated content and human-written content. One of the main challenges of this problem is that AI-generated text is changing rapidly as models improve. The advent of the most recent generative GPT models, such as ChatGPT 4, Mistral, Llama 2, etc. mean that the field is constantly evolving.

Bayesian approaches may handle this challenge better than classical approaches, because they can update its beliefs (i.e., the posterior distribution) with new data using Bayes' theorem. This means that as new data comes in, the Bayesian model can adjust its parameters and predictions can change, without discarding the previous information. A classical approach, on the other hand, would need to re-estimate the parameters and re-train the model every time new data is available, which can be inefficient and prone to overfitting. This might also lead to more advanced Bayesian models that can adjust on the fly as data is streamed into them.

Another advantage of the Bayesian approach is that it can provide a measure of uncertainty for its predictions. This is important because some texts might be more ambiguous or difficult to classify than others. Bayesian approach can give us not only the most likely label (AI-generated or human-written), but also the probability of that label being correct. This can help us assess the confidence and reliability of our predictions, and identify the cases where we need more information or human intervention. Classical approach, on the other hand, would only give us a point estimate, without any indication of how certain or uncertain it is. This sis especially helpful when dealing with issues of fake news, AI generated scams or even in academic settings, where it's important to be vigilante about checking content, however, we cannot just assume a model is correct - we need to be wary of it's uncertainty (additionally because large language models of probabilistic).

Research

The research of text generated by Artificial Intelligence (AI) and text written by humans is an area of growing interest, especially with the explosion of LLMs such as GPT-4, Llama, etc.. Researchers use a variety of methods to distinguish between AI-generated and human-written text. These can be categorized into three approaches: linguistic analysis, statistical methods, and machine learning techniques.

Linguistic Analysis: This approach analyzes the text for linguistic features that are characteristic of either human or AI writing. For example, this might entail looking at syntactical patterns for each label. Analytics look for various patterns such as repetitiveness, complexity, and the use of cliche language. One instance might be that AI written text exhibits higher degrees of repetitiveness or a lack of understanding of context-specific idioms. On the other hand, human writing often contains inconsistencies and errors that AI models typically do not make, owing to their training on vast datasets of edited text.

Statistical Methods: These methods entail parsing the statistical properties of the text. For example, researchers might look at the distribution of word frequencies: tf–idf, sentence lengths, or the use of certain types of words (like pronouns or conjunctions). AI-generated text, for example, might show a different statistical signature compared to human writing due to the way AI models learn and generate text.

Machine Learning Techniques:This entails using machine learning models to classify text as AI-generated or human-written. These models are trained on large datasets containing examples of both types of text. They learn to identify subtle patterns and features that might not be easily found by human analysis. Deep learning models, such as using neural networks, have been found to be particularly effective in this domain. However, the challenge lies in continually updating these models (they are resource heavy and can be expensive) as AI text generation technology evolves, making it a dynamic and ongoing area of research.

One interesting method is watermarking. Researchers are working on AI generation models that can embed hidden signals in the text, such as a particular frequency of words (or lack of words) as a means of detecting AI generated text.

Bayesian Approach

Utilizing PyMC, I approach this problem, aiming to create a model for Bayesian analysis. This model is grounded in probabilistic modeling.

First I use **Feature Extraction with TF-IDF**: This first step involves transforming the text data into a numerical format that can be analyzed. I used the Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer, which converts the text into a matrix of TF-IDF features. I set max_features=500 (however, this can be far greater - as many words in the document) in order to speed up the process.

Here is how this works:

TF-IDF(t, d) = TF(t, d) * IDF(t)

TF(t, d) = (number of times t appears in d) / (total number of tokens in d)

IDF(t) = log [(total number of documents) / (number of documents with t in it)]

TF stands for term frequency, which is the ratio of the word count to the total number of words in the document. IDF stands for inverse document frequency, which is the logarithm of the inverse of the proportion of documents that contain the word.

My **Data Preparation**: Next I split the data into training and testing sets, ensuring that both the features (X) and labels (y) are correctly aligned. The labels in my case are binary (generated), indicating whether the text is AI-generated or human-written. I used 5000 essays labeled 0, or 1 (if they were AI generated) from https://www.kaggle.com/datasets/thedrcat/daigt-external-train-dataset. This dataset was found via the Kaggle competition for determining AI generated text: LLM - Detect AI Generated Text | Kaggle

	generated	text
5525	0	Does working on a school project during your s...
20487	0	Dear Senator,\n\nI believe that the United Sta...
26827	0	Sometimes when you are in a tough spot you see...
4246	0	Did you know that passenger cars are responsib...
28638	1	I believe that students should have the right ...

A snapshot of the daigt training data of AI generated labeled (1) essays and human written essays.

I then converted the sparse matrix of vectorized text to a **Dense Matrix**: PyMC struggles with sparse data. This was necessary for processing in PyMC.

Building the **Bayesian Model**: in my Bayesian model, I defined prior distributions for the parameters alpha (the intercept) and beta (the coefficients for each TF-IDF feature). The choice of normal distributions with specified means (mu) and standard deviations (sigma) reflects my initial assumptions about these parameters are built from my initial beliefs about this data. The alpha parameter (that I used) has a tighter prior (smaller sigma), which points to less variability, while beta has a looser prior (larger sigma), allowing for more variability across the features (since there are so many features in a corpus of text – each word).

My priors:

```
alpha = pm.Normal('alpha', mu=0, sigma=5):
```

This is a normal distribution with a mean (mu) of 0 and a standard deviation (sigma) of 5. It represents my beliefs about alpha before seeing the data. I aimed here (as a non-expert in this field) to share a belief that there is no inherent bias towards AI-generated or human-generated text (i.e., the odds are even when all features are zero). In further consideration, I might have chosen an alpha that slightly favored the human text – as there were more human generated texts than AI in our data.

```
beta = pm.Normal('beta', mu=0, sigma=20,
shape=X_train_dense.shape[1]):
```

This is also a normal distribution and a prior distribution for the variable beta. It has a mean of 0 and a standard deviation of 20. The shape parameter indicates that beta is a vector of length X_train_dense.shape. Since the standard deviation is larger, this is a more open prior, meaning I am less certain about the value of beta.

$\alpha \sim N(0,52)$

$\beta \sim N(0,202)$

My **likelihood function** is the likelihood of the observed data (`y_train`) is modeled as a Bernoulli process, with the probability p of a text being AI-generated being modeled as a sigmoid function of the linear combination of TF-IDF features and their corresponding beta coefficients, plus the intercept alpha. This allows for uncertainty in the parameters, and using the sigmoid function, it handle binary outcomes, ensuring the predicted probabilities are between 0 and 1

$p=1+e-(\alpha+X \cdot \beta)1$

$y \sim Bernoulli(p)$

Using PYMC I perform inference using **Markov Chain Monte Carlo (MCMC)** sampling. This process creates samples from the posterior distribution of my model parameters (alpha and beta) given the observed data.

After running the MCMC simulation, I further analyze the 'trace'. This analysis will help me understand which features (words) are most likely of AI-generated versus human-written text, as well as the overall likelihood of a piece of text being AI-generated.

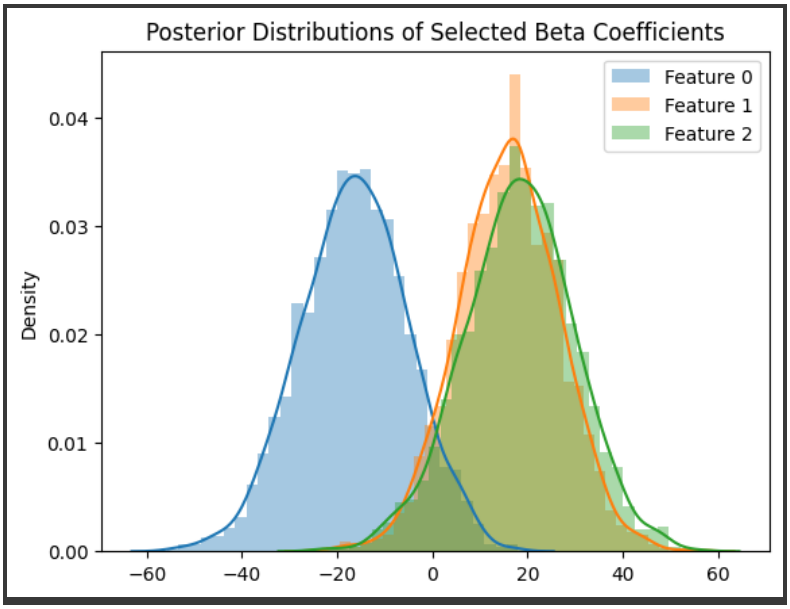
Analysis

AI-Indicative Words:	**Human-Indicative Words:**
word importance 96 conclusion 58.785009 335 potential 46.197861 344 provide 43.908054 219 important 41.154936 165 finally 39.892766 25 and 36.206217 377 significant 35.585662 349 re 35.545914 58 benefits 35.476974	word importance 50 because -68.952490 352 reason -45.035263 140 electors -43.875666 262 many -42.055899 461 very -41.637847 466 voting -37.400886 20 although -37.153518 184 going -34.275403 320 percent -34.068907

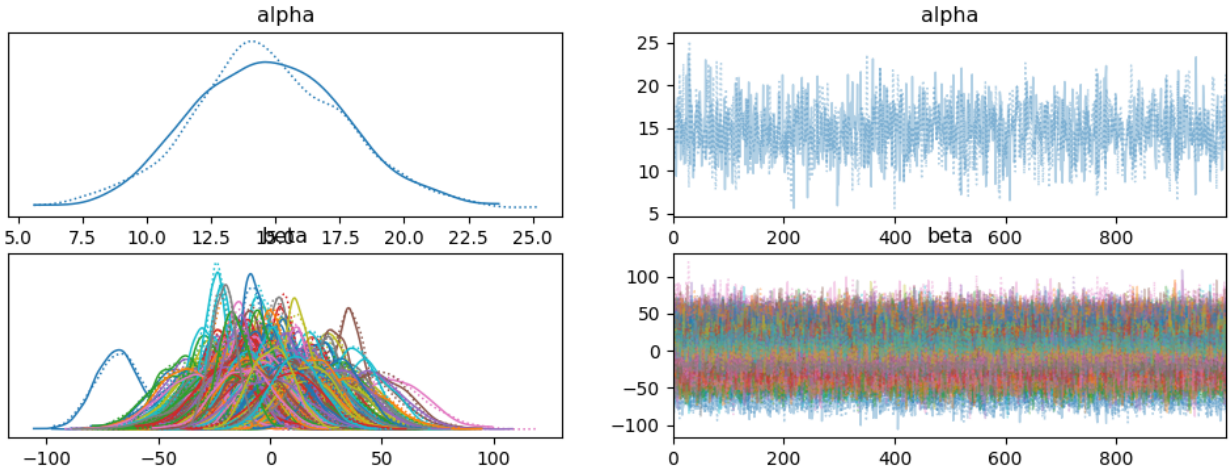
480	while	33.248029	192	had	-32.439397

The model output displayed the most important word scores to predict a 1 (AI generated) with the word 'conclusion' coming out on top and the least likely used word for this AI text as 'because'.

Graphing the top three feature words:



This shows the posterior distribution of the top three features.

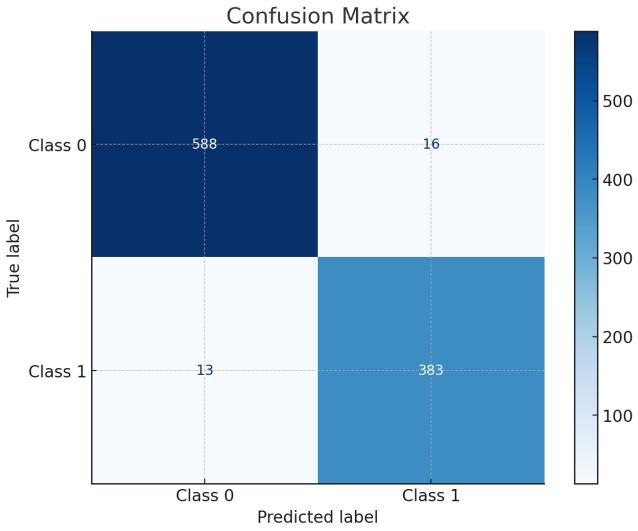


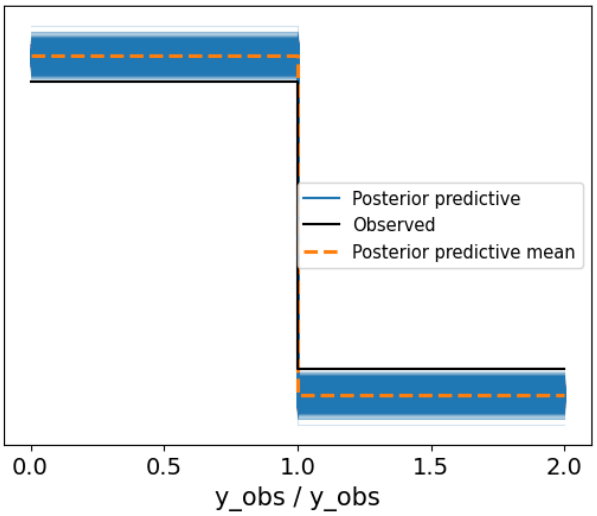
These four graphs show the alpha and beta distributions in more detail. Using multiple overlaid kerbal dentistry estimates of different distributions. The right graph shows the convergence of a MCMC (with a 'fuzzy caterpillar' looking – without a visible trend).

In analyzing the model, the accuracy, precision and recall were remarkably high. This is likely pointing to some data leaking or overfitting inside the model.

Prediction Results:
Accuracy: 0.971
Precision: 0.9598997493734336
Recall: 0.9671717171717171
F1 Score: 0.9635220125786164
ROC-AUC Score: 0.9965925814435748

Confusion Matrix:





This is a Kernel Density Estimate graph to show the probability density function of a random variable. It visualizes the distribution of data and shows a good fit between observed and predicted.

Challenges

The high scores are likely caused by some errors in my model.

One possible issue is overfitting: Because there are so many features, a possible issue with Monte Carlo simulation is overfitting due to so many features. I might have incorrectly specified my priors and could aim to loosen them in the future.

There is possibly Bias in the data. There were two times as many human generated texts in the data than AI created ones.

Future Improvements

More data should be used in the future (especially AI generated text) to even out the training. In addition, better priors should be used (allowing for more space for the interpretation).

The dataset I used was only a portion of the over a gigabyte of essays collected by the data engineer Derek Kleczek (see references). This provides ample opportunity for a deeper analysis. In pursuing other projects in the Kaggle Competition, no one else was using a Bayesian approach. A combined approach using classifical statistics, Bayesian Analysis and deep neural networks could provide an even more robust analysis that what is presented in this paper.

Conclusion

In summary, my model employed a Bayesian approach using PyMC and TF-IDF for feature extraction. I chose priors, then used a bernoulli likelihood along with a sigmoid function. This approach falls under the broader category of statistical methods, with a specific focus on probabilistic modeling and Bayesian inference.Data:

References

Chen, Y., Li, Y., Narayan, R., Subramanian, L., & Xie, B. (2021). DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. Journal of Machine Learning Research, 22(114), 1-49. Retrieved from [https://jmlr.csail.mit.edu/papers/volume22/18-332/18-332.pdf]

Gehrmann, Sebastian, Hendrik Strobelt, and Alexander M. Rush. “GLTR: Statistical Detection and Visualization of Generated Text.” In Proceedings of the 57th Annual Meeting of the

Association for Computational Linguistics: System Demonstrations, 111–116. Florence, Italy: Association for Computational Linguistics, 2019.

“Handwriting Recognition with ML (An In-Depth Guide).” Nanonets. Last modified September 14, 2023.

Kaggle. 2023. LLM - Detect AI Generated Text. Retrieved from [https://www.kaggle.com/competitions/llm-detect-ai-generated-text]

“Linguistic Analysis - Discourse Analyzer AI Toolkit.” Discourse Analyzer. Accessed April 5, 2023.

TheDRCat. 2023. DAIGT External Train Dataset [Data set]. Kaggle. Retrieved from [https://www.kaggle.com/datasets/thedrcat/daigt-external-train-dataset]