VLG Open Project Proposal 2023

AI-Text Detection

Aman Behera | Chemical Engineering | 22112013 | aman_b@ch.iitr.ac.in

Overview:

In recent years, large language models (LLMs) have become increasingly sophisticated, capable of generating text that is difficult to distinguish from human-written text. In this project. I hope to foster open research and transparency on AI detection techniques applicable in the real world. As LLMs continue to expand, how can we detect LLM-generated texts to help minimize the threat posed by the misuse of LLMs?

The course of action, Insights, and understanding the working of BERT behind the prevails of code:

Here, I have noted the course of action (tentative) of how I would be proceeding with the pipeline and how does BERT doe feature selection, how BERT classifies texts into human-authored and AI-generated texts, I also aimed to explain what are the different ways of AI-text detection namely the white-box and black-box detection. Also, how BERT takes inputs, and how it can be improved have been covered so as to give a complete 360-degree overview of the hows, whys, and whats.

Analysis of the problem statement and course of action: https://shorturl.at/imyR7

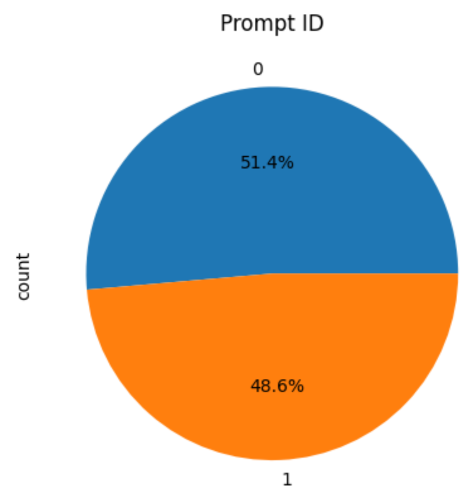
Video explanation of my analysis and tentative course of action and explanation of how BERT extracts features and what happening behind the codes to get ground insights upon how BERT does classification: https://shorturl.at/hpsD3, https://shorturl.at/afmnV

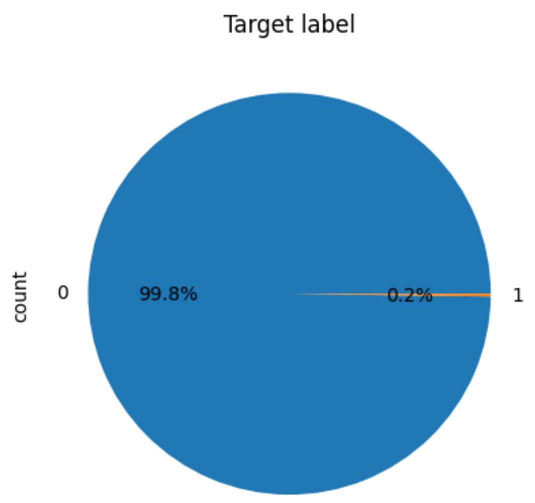
Still, let's have a summary overview of Project Specifications here Milestones:

- **Data PreProcessing & Cleaning -** As per the dataset I feel using prompt id as the base would be better as using 'generated' as the base of the model classification won't work well as the data is an imbalance concerning the 'generated' column (98:2). In contrast, using prompt-id it would give better results because as per it the distribution is (57:43) (kinda better). And also no columns have NaN values so we don't have to dropna()

This is the data frame of the dataset and we can see that there are no NaN-values

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1378 entries, 0 to 1377
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   id         1378 non-null   object
 1   prompt_id  1378 non-null   int64
 2   text       1378 non-null   object
 3   generated  1378 non-null   int64
```





The above is for prompt-id-based classification of training data and another is based on the 'generated' column.

- **Removing StopWords, Tokenizing, and Removing punctuations:** splitting the words based on spaces and forming a list out of it and then removing the stop words like (it, then, would, etc) from it and doing the usual process of converting the usual process of converging different words with similar meaning to the same would (like loved, loving, love would have the same base word love, etc) also known as a lamentation.
- **using Bert-Preprocessing model-** we can do the above steps by the usage of nltk library and then using techniques like word2vec or bag of words for vectorization, but using the bert-preprocess trained model would give better results which can be accessed from kaggle→notebook→model→bert→un-en-preprocess, `/kaggle/input/bert/tensorflow2/en-uncased-preprocess/3`
- **Model building -** again we can use a pre-trained bert-model from the kaggle tensorflow collaboration and use it in our Kaggle notebook, for model building, but we need to have tensorflo_hub preinstalled (it didn't work in Jupyter or Google Collab the time I tried, but it worked on kaggle notebook)

`/kaggle/input/bert/tensorflow2/bert-en-uncased-l-12-h-768-a-12/2`

- **Regularisation -** using a dropout layer following it to ease out overfitting
- **Dense layer -** adding a dense layer following it with 1 output as we want either 0/1 (binary classification problem).
- **Compile -** model.compile() our model, with optimizer as adam as its the best (tho I feel like rmse would do better as in shorter epochs, rmse generally tends to be better but will see), and with loss as `'binary_crossentropy'`,
- **Fitting -** define our batch_size, and set the epochs, I read from various papers that Bert model doesn't require many epochs as it's a very tightly built model, so I feel we should start with something as minimal as 5-10.
- **Competition Requirements -** we then save our trained model and then use it to predict and then save the outputs in submission.csv which would be then submitted.

*I also have a **street-smart idea** of how we can make our model training more powerful i.e. why only rely on the dataset given in the Kaggle competition? more such similar competitions are also there and I made their list they also have essays and the idea was to detect AI-generated texts so, we would improve our training dataset from* **1378 to around 60k!!!!! (we would first select the appropriate columns from each dataset and then form a new data frame for all and then concat all)**

They are as follows 😀

train_drcat_04.csv
train_essays_RDizzl3_s
LLM_generated_essay_P
falcon_180b_v1.csv
llama_70b_v1.csv

(I have included them in my drive folder of my submission)

Week I (Data Preprocessing and Cleaning) :

- Data cleaning, pre-processing, and EDA
- Removing the stop words, lowercase conversion of all the subsequent words, removing the punctuation, and non-alphabetical words and then Tokenizing, applying padding,
- Converting the tokenized list into tensors as BERT needs the inputs to be in this state
- Also, I would be trying the variation in which I won't be lowercasing all the words and would be using 'bert-large-cased' instead of 'bert-base-uncased' as it does differentiate between the casing of letters (i.e. english and English would be different for the former) hence I believe it might come up as a helping parameter which can help in classifying the texts as humans sometimes make mistakes in the casing, unlike LLMs.

Week II (BERT training and Prediction) :

- Training the BERT model and using the trained model to make predictions
- Evaluating how well it did, using AUC scores and the efficiency formula given in the problem statement.
- Saving the result in CSV file in the format asked in the problem statement

Week III (Focus on Fine Tuning and Visualisation):

- Learning what are all the hyperparameters of the BERT model, and using efficient search algorithms to find the best fit.
- Storing the epoch results in a variable and plotting the results.
- Subsidiary plots would also be made to get more insights into the training process, the data distribution, and the results procured.

Week IV (increasing the readability of code by more comments & setting up on Github) :

- Increasing the readability of code by adding more comments and subsequent documentation would be done on medium.
- Then the whole project would be uploaded on Github and if time persists I would also record a video of me explaining what I did, how I did & why I did.
- If time persists I would also try further variations of BERT (its sister models basically) and see how well they perform.

Two cents about Me

Aman Behera: Hello guys, I am a sophomore from the Chemical Engineering dept and an upcoming winter **ML winter research intern** at ***XLRI Jamshedpur*** where I wish to work on **time series analysis**. I have also taken up a research intern project (remote) under a professor at ***IIM Kashipur*** in the domain of ML in finance. Currently, I spend time practicing and learning machine learning and I love modeling and forecasting. I also did a project under ***Prof. Anshu Anand*** from the ***Chemical dept*** in analyzing particle orientation using OpenCV, Recently I completed the ***Monte Carlo Simulation open project*** by the finance club. I also like to do competitive programming and lay my hands on various platforms on and off. I also like writing tech blogs in my free time in the medium. I like to do sketching as a hobby and watch underrated mystery movies.
My GitHub, Medium, LinkedIn.

Contact Details

- Aman Behera
- 22112013
- aman_b@ch.iitr.ac.in
- 9587987718