CSCI6364 - Machine Learning

Final Report

Project Title:

Multigenerator, Multidomain, and Multilingual Machine-Generated Text Detection

By:

Dandan Chen
Satish Sarma
Mohit Reddy

Index

1. Introduction

a. Overview

Large language models (LLMs), such as Chat-GPT have become increasingly powerful in generating human-like text. These models are trained on vast amounts of diverse data and are able to generate coherent and contextually relevant responses to requests made by humans. While these models have shown remarkable capabilities in various natural language processing (NLP) tasks, they have also raised concerns about the potential misuse of their abilities. One significant concern is the generation of machine-generated text that is intended to mimic human writing to the point of being indistinguishable from genuine human-produced content. Large language models can be used to automatically generate articles, essays, create convincing phishing emails, messages, or social media posts, fake reviews, comments, or feedback, etc.

b. Problem Definition and Understanding

To counter the misuse of LLMs in this way, there is a growing need for the development of robust and effective methods to detect human-written text from machine-generated text. *This project aims to do just that, to train and optimize a Machine Learning model for detecting whether a given piece of text is human-written or machine-generated, and predicting which large language model has generated the text, if it happens to be machine-generated.* By training models to recognize the distinctive patterns, styles, or features associated with machine-generated text, we can potentially make it possible to create safeguards against the misuse of language models. The development of such detection models involves careful consideration of linguistic features, context understanding, and the evolving nature of language models themselves.

2. Project Data and Presentation

a. Project Data

Our project has been chosen from Topic no.8 of SemEval2024 tasks. The dataset used is an extension of the M4 dataset[1] described in the paper with the same title[2]. The dataset is in JSON format and consists of a collection of sentences, categorized into human-written and machine-generated text. The machine-generated text is further categorized into 5 labels, indicating which one of the five LLMs - Davinci003, ChatGPT, Cohere, Dolly-v2 or BLOOMz, has generated the text. The total number of data values (a.k.a labeled text) in this dataset is 133,551. Further statistics of the dataset have been displayed in the SemEval task8 Github repo[3], also shown in the figure below:

Source/ Domain	Language	Total Human	Parallel Data						
			Human	Davinci003	ChatGPT	Cohere	Dolly-v2	BLOOMz	Total
Wikipedia	English	6,458,670	3,000	3,000	2,995	2,336	2,702	3,000	17,033
Reddit ELI5	English	558,669	3,000	3,000	3,000	3,000	3,000	3,000	18,000
WikiHow	English	31,102	3,000	3,000	3,000	3,000	3,000	3,000	18,000
PeerRead	English	5,798	5,798	2,344	2,344	2,344	2,344	2,344	17,518
arXiv abstract	English	2,219,423	3,000	3,000	3,000	3,000	3,000	3,000	18,000
Baike/Web QA	Chinese	113,313	3,000	3,000	3,000	–	–	–	9,000
RuATD	Russian	75,291	3,000	3,000	3,000	–	–	–	9,000
Urdu-news	Urdu	107,881	3,000	–	3,000	–	–	–	9,000
id_newspapers_2018	Indonesian	499,164	3,000	–	3,000	–	–	–	6,000
Arabic-Wikipedia	Arabic	1,209,042	3,000	–	3,000	–	–	–	6,000
True & Fake News	Bulgarian	94,000	3,000	3,000	3,000	–	–	–	9,000
Total			35,798	23,344	32,339	13,680	14,046	14,344	133,551

b. Project structure

Our work consists of Two objectives/subtasks

1. **Subtask A -** ***Binary Human-Written vs. Machine-Generated Text Classification:*** Given a full text, our objective is to make our model predict whether it is human-written or machine-generated. There are two tracks for subtask A: monolingual (only English sources) and multilingual.

2. **Subtask B - Multi-Class Machine-Generated Text Classification:** Given a full text, the objective is to make our model predict who generated it. It can be human-written, or generated by a specific language model (one of the 5 models).

We train our model three times, twice for each of the monolingual/multilingual datasets of subtask A, and a third time for subtask B. We perform our predictions on the development set data which was already provided in the downloaded dataset. The *format_checker.py* modules in each subtask verify that your prediction file complies with the expected format - one single JSONL file for all texts, and the entry for each text must include the fields "id" and "label". The *scorer.py* module contains the scorer which will report the official evaluation metric and other metrics for a given prediction file.

The **official evaluation metric** for both Subtask A and Subtask B is **Accuracy**. However, the scorer also reports macro-F1 and micro-F1.

3. Project Description and Methods

a. Preprocessing

The preprocessing technique used here is tokenization. Tokenization is the process of breaking down a text into individual units called ‘tokens’. In the context of natural language processing (NLP) with transformers, tokens are typically words, subwords, or characters. In our code, we create a function “*preprocess_function”* which is applied to the training and test datasets using the ‘map’ function:

1. It takes a batch of examples, where each example has a "text" field.
2. It applies tokenization using the tokenizer specified by us. In this case, we have used the default tokenizer in the ‘transformers’ dataset.
3. The tokenized sequences are truncated to fit the maximum length allowed by the model if they exceed that length.

The input text data is tokenized appropriately for both training and evaluation datasets before being used to fine-tune the model or make predictions. The tokenizer is loaded from a pretrained model using the “*AutoTokenizer.from_pretrained*” method. We use the ‘transformers’ dataset library for this.

b. Machine Learning Model

For our training, we use the XLM-RoBERTa model, which is a part of the Transformers library by Hugging Face. XLM-RoBERTa stands for "Cross-lingual Language Model - RoBERTa," and is well suited to handle multilingual tasks.

1. The model is an extension of the BERT (Bidirectional Encoder Representations from Transformers) model. It aims to improve upon BERT's performance by optimizing various hyperparameters, training data size, training duration, and other training methodologies.

2. The model is designed to handle multiple languages, making it useful for tasks that involve text in different languages. It learns to understand and represent text in multiple languages effectively.

3. The model, like BERT, is based on the transformer architecture, which comprises multiple layers of self-attention mechanisms. This architecture allows it to capture dependencies and relationships between words or tokens in text data.

c. Prediction

For subtask A, we perform binary classification and for subtask B, we perform multi-class classification. We define several functions in our code for each of our required purposes:

- *Preprocess_function*: Tokenizes input text data using the provided tokenizer.
- *Get_data*: Reads and preprocesses training and test datasets.
- *Compute_metrics*: Computes evaluation metrics for the model.
- *Fine_tune*: Fine-tunes the model on the training data.
- *Test*: Evaluates the fine-tuned model on the test data and returns results.

We use the command line to run the script, specifying the paths to training and testing files, the subtask, and the output prediction file. The prediction is done in the '*test*' function, which evaluates the fine-tuned model on the test dataset. The *softmax* function is applied to get probabilities, and the predictions are obtained by selecting the class with the highest probability.

The script loads a classification report using the *evaluate* library and computes the results by comparing the predicted labels (*preds*) with the true labels (*p[rediction_labels.ids*). The results are returned as a dictionary, and the predictions are also returned for further analysis in the *scorer* module.

Execution

note: The execution of the model can be seen in the video presentation. Due to the size of the dataset, it takes upto 10 hours to train the model for each of the three datasets. For the video presentation, we have chosen a small part of the data so that we can show the entire execution in the video. This resulted in the high scorer results for subtask A track 1. For the report, we have extensively trained our model on the entire data to obtain more accurate results.

4. Scorer and Analysis of Results

The scorer evaluates the results of the model using accuracy, macro-F1 and micro-F1 scores. Micro and macro F1 scores are variants of the F1 score, which is a metric commonly used to evaluate the performance of a classification model. The F1 score is the harmonic mean of precision and recall.

1. *Accuracy* - It is calculated as the ratio of correctly predicted instances to the total number of instances in the dataset.
2. *F1 Score* - The F1 score is the harmonic mean of precision and recall. Micro F1 calculates precision, recall, and F1 score globally across all classes, whereas Macro F1 calculates precision, recall, and F1 score for each class independently.

The code consists of the function 'evaluate' which does this evaluation, and a function 'validate_files' that checks the format of the prediction file using the function from the format_checker.py module. The results after execution are shown below:

Results:

1. **Subtask A**

Monolingual text - Training

```
{'train_runtime': 893.7245, 'train_samples_per_second': 8.056, 'train_steps_per_second': 0.504, 'train_loss': 2.9780612223678165e-06,
100% 450/450 [14:53<00:00,  1.99s/it]
Map: 100% 500/500 [00:00<00:00, 956.72 examples/s]
You're using a XLMRobertaTokenizerFast tokenizer. Please note that with a fast tokenizer, using the `__call__` method is faster than
100% 63/63 [00:14<00:00,  4.30it/s]
Downloading builder script: 100% 5.24k/5.24k [00:00<00:00, 15.3MB/s]
```

Monolingual Text - Evaluation

```
!python • subtaskA/scorer/scorer.py • --gold_file_path=subtaskA/Monoling

INFO : Prediction file format is correct
INFO : macro-F1=1.00000 micro-F1=1.00000        accuracy=1.00000
```

Multilingual text - Training

```
{'eval_loss': 0.00030841672560200095, 'eval_f1': 1.0, 'eval_runtime': 33.1187, 'eval_samples_per_second':
100% 750/750 [21:59<00:00,  1.56s/it]
100% 63/63 [00:32<00:00,  2.31it/s]
{'train_runtime': 1347.9121, 'train_samples_per_second': 8.903, 'train_steps_per_second': 0.556, 'train_l
100% 750/750 [22:27<00:00,  1.80s/it]
Map: 100% 4000/4000 [00:12<00:00, 331.96 examples/s]
You're using a XLMRobertaTokenizerFast tokenizer. Please note that with a fast tokenizer, using the `__ca
100% 500/500 [01:40<00:00,  4.96it/s]
Downloading builder script: 100% 5.24k/5.24k [00:00<00:00, 19.1MB/s]
```

Multilingual Text - Evaluation

```
!python  subtaskA/scorer/scorer.py  --gold_file_path=subtaskA/Multil

INFO : Prediction file format is correct
INFO : macro-F1=0.61669 micro-F1=0.65050        accuracy=0.65050
```

2. Subtask B

Training:

```
{'eval_loss': 0.0067004249431192875, 'eval_f1': 0.9983333333333333, 'eval_runtime': 23.2071, 'eval_samples_per_second': 25.854, 'eval_steps_per_second': 1.637,
100% 450/450 [14:39<00:00,  1.69s/it]
100% 38/38 [00:22<00:00,  2.09it/s]
{'train_runtime': 907.8373, 'train_samples_per_second': 7.931, 'train_steps_per_second': 0.496, 'train_loss': 0.04395314534505208, 'epoch': 3.0}
100% 450/450 [15:07<00:00,  2.02s/it]
Map: 100% 1000/1000 [00:01<00:00, 628.47 examples/s]
You're using a XLMRobertaTokenizerFast tokenizer. Please note that with a fast tokenizer, using the `__call__` method is faster than using a method to encode th
100% 125/125 [00:35<00:00,  3.53it/s]
```

Evaluation:

```
!python  subtaskB/scorer/scorer.py  --gold_file_path=subtaskB/subt

INFO : Prediction file format is correct
INFO : macro-F1=0.52216 micro-F1=0.59900        accuracy=0.59900
```

Analysis of Results:

1. We observe around 59% accuracy in prediction by the model. This indicates that the model is decent in predicting whether the text is human-written or AI generated during multi-class classification.

2. However, in the case of binary classification, the model appears to perform much better, with high scores in both monolingual text prediction and multilingual prediction

5. Conclusion

This report focused on detecting human-written vs. machine-generated text in the context of SEMEVAL Task 8, Subtask A and B. The implementation utilized a transformer-based model, specifically the XLM-RoBERTa architecture, and involved a comprehensive pipeline including data preprocessing, model training, and evaluation. The developed model achieved a commendable accuracy of around 60-65% and an F1 score of 0.6 on average. These metrics indicate the model's effectiveness in distinguishing between human and machine-generated text. Further tuning and refining of the model by means such as cross-validation can improve the model's performance. Future work on this model can include adding more LLMs to the dataset to increase the scope of predicting, increasing the size of the dataset to improve model accuracy, and potentially designing or customizing our own custom neural network architecture tailored to the specifics of our task.

6. References

- https://github.com/mbzuai-nlp/M4
- arXiv:2305.14902 [cs.CL]
- https://github.com/mbzuai-nlp/SemEval2024-task8
- https://huggingface.co/docs/transformers/index