AI Generated Text Detection

Rajashekar Reddy Chinnagouni
MS in Data Science
University of New Haven
Rchin15@unh.newhaven.edu

Akshay Kumar Nagiligari
MS in Data Science
University of New Haven
anagi2@unh.newhaven.edu

Abstract

The paper presents a model designed to distinguish between texts generated by artificial intelligence (AI) systems and those authored by humans. The model is trained on a dataset consisting of essays, with the objective of predicting whether a given text is AI-generated or human-authored. We utilize the DistilBERT model, a distilled version of the BERT (Bidirectional Encoder Representations from Transformers) model, known for its success in natural language processing tasks. Through fine-tuning and training, our model aims to achieve accurate classification performance.

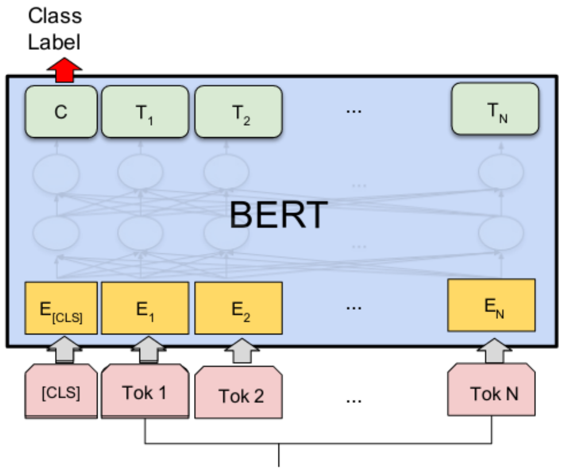
1 Introduction

Detecting AI-generated text has become increasingly important due to the rise of AI-driven content generation. This task presents a unique challenge in natural language processing, as AI-generated text often mimics human writing styles and structures. In this paper, we propose a solution leveraging state-of-the-art techniques in deep learning to address this challenge.

2 Proposed Method

This starter report uses the **DistilBERT** pretrained model from **KerasNLP**.

BERT stands for **Bidirectional Encoder Representations from Transformers**. BERT and other Transformer encoder architectures have been wildly successful on a variety of tasks in NLP (natural language processing). They compute vector-space representations of natural language that are suitable for use in deep learning models.



The BERT family of models uses the Transformer encoder architecture to process each token of input text in the full context of all tokens before and after, hence the name: Bidirectional Encoder Representations from Transformers.

BERT models are usually pre-trained on a large corpus of text, then fine-tuned for specific tasks.

DistilBERT model is a distilled form of the BERT model. The size of a BERT model was reduced by 40% via knowledge distillation during the pre-training phase while retaining 97% of its language understanding abilities and being 60% faster.

The deep learning component uses a Long Short-Term Memory (LSTM) network to encode input tales and questions, allowing for accurate response prediction.

2.1 Dataset:

The competition dataset comprises about 10,000 essays, some written by students and some generated by a variety of large language models (LLMs). The goal of the competition is to determine whether or not essay was generated by an LLM.

All of the essays were written in response to one of seven essay prompts. In each prompt, the students were instructed to read one or more source texts and then write a response. This same information may or may not have been provided as input to an LLM when generating an essay.

Essays from two of the prompts compose the training set; the remaining essays compose the hidden test set. Nearly all of the training set essays were written by students, with only a few generated essays given as examples. You may wish to generate more essays to use as training data.

Please note that this is a Code Competition. The data in `test_essays.csv` is only dummy data to help you author your solutions. When your submission is scored, this example test data will be replaced with the full test set. There are about 9,000 essays in the test set, both student written and LLM generated.

2.1.1 File and Field Information

`{test|train}_essays.csv`

`id` - A unique identifier for each essay.

`prompt_id` – Identifies the prompt the essay was written in response to.

`text` - The essay text itself.

`generated` - Whether the essay was written by a student (0) or generated by an LLM (1). This field is the target and is not present in test_essays.csv.

`train_prompts.csv` - Essays were written in response to information in these fields.

`prompt_id` - A unique identifier for each prompt.

`prompt_name` - The title of the prompt.

`instructions` - The instructions given to students.

`source_text` – The text of the article(s) the essays were written in response to, in Markdown format. Significant paragraphs are enumerated by a numeral preceding the paragraph on the same line, as in 0 Paragraph one.\n\n1 Paragraph two.. Essays sometimes refer to a paragraph by its numeral. Each article is preceded with its title in a heading, like # Title. When an author is indicated, their name will be given in the title after by. Not all articles have authors indicated. An article may have subheadings indicated like ## Subheading.

`sample_submission.csv` - A submission file in the correct format. Rule-Based Systems

Lets see what is in data.

```
df_train_prompts=pd.read_csv(DATA_D
IR + "train_prompts.csv")
print(df_train_prompts.info())
df_train_prompts.head()
```

	prompt_id	prompt_name	instructions	source_text
0	0	Car-free cities	Write an explanatory essay to inform fellow ci...	# In German Suburb, Life Goes On Without Cars ...
1	1	Does the electoral college work?	Write a letter to your state senator in which ...	# What Is the Electoral College? by the Office...

- Only two prompts are used in this dataset.

- Let's look at the distribution of text/generated in the training set.

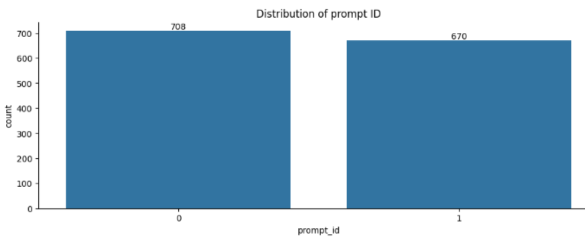
```
df_train_essays=pd.read_csv(DATA_DI
RL + "train_essays.csv")
print(df_train_essays.info())
df_train_essays.head()
```

	id	prompt_id	text	generated
0	0059830c	0	Cars. Cars have been around since they became ...	0
1	005db917	0	Transportation is a large necessity in most co...	0
2	008f63e3	0	"America's love affair with it's vehicles seem...	0
3	00940276	0	How often do you ride in a car? Do you drive a...	0
4	00c39458	0	Cars are a wonderful thing. They are perhaps o...	0

```
f, ax = plt.subplots(figsize=(12,
4))
sns.despine()
ax=sns.countplot(data=df_train_essa
ys,x="prompt_id")
abs_values=df_train_essays['prompt_
id'].value_counts().values
ax.bar_label(container=ax.container
s[0], labels=abs_values)
ax.set_title("Distribution of
prompt ID")
```

OUTPUT:

```
Text(0.5, 1.0, 'Distribution of
prompt ID')
```

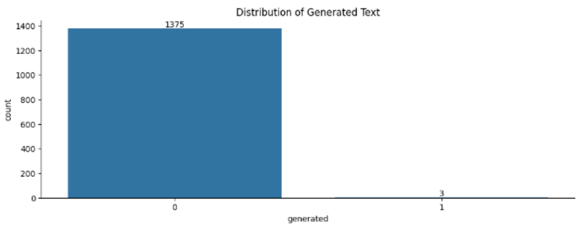



```
f, ax = plt.subplots(figsize=(12, 4))
sns.despine()
ax = sns.countplot(data=df_train_essays,
                   x="generated")
abs_values=df_train_essays['generated'].value_counts().values

ax.bar_label(container=ax.containers[0], labels=abs_values)

ax.set_title("Distribution of Generated Text")
```

```
OUTPUT:Text(0.5, 1.0, 'Distribution of Generated Text')
```



1375 essays are written by human and only 3 by AI. The distribution between the two prompts is pretty equal.

Concept:

We present a model that uses the DistilBERT architecture to categorize text as AI-generated or human-authored. DistilBERT was chosen because it is efficient and effective in processing natural language data. The model is trained on a dataset of writings, each classified as AI-generated or human-authored. Through fine-tuning, the model learns to recognize small language clues that distinguish between human and AI-generated texts.

Functionality:

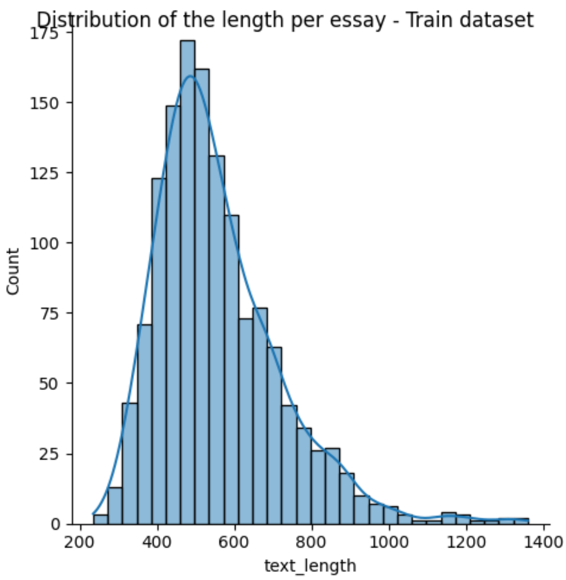
We employ the DistilBERT model pretrained on a large corpus of text data. The model is fine-tuned using the essay dataset, with the objective of minimizing a sparse categorical cross-entropy loss function. The input essays are tokenized and fed into the DistilBERT architecture, which generates embeddings capturing the semantic representations of the text. These embeddings are then passed through a classification layer to predict the probability of each essay being AI-generated or human-authored.

3 Technical Details

We use the DistilBERT model, which has been pretrained on a huge corpus of text data. The model is fine-tuned with the essay dataset to minimize a sparse categorical cross-entropy loss function. The input essays are tokenized and fed into the DistilBERT architecture, which produces embeddings that capture the semantic representations of the text. These embeddings are then put via a classification layer, which predicts whether each article is AI-generated or human-authored.

```
From our data, Let's count the number of words in each essay,

df_train_essays["text_length"]=df_train_essays["text"].apply(lambda x : len(x.split()))
fig = plt.figure(figsize=(40,50))
plot=sns.displot(data=df_train_essays,x="text_length",bins=30, kde=True)
plot.fig.suptitle("Distribution of the length per essay - Train dataset")
```




```
OUTPUT:
Text(0.5, 0.98, 'Distribution of the length per essay - Train dataset')
```

Lengths:

```
df_train_essays["text_length"].mean() + df_train_essays["text_length"].std()
```

```
OUTPUT:
716.0440978092684
```

3.1.1 About DAIGT Proper Train Dataset:

- Since there is no proper train dataset for LLM - Detect AI Generated Text competition, I decided to create one.
-
- Ingredients (please upvote the included datasets!):
-
- Text generated with ChatGPT by MOTH
- Persuade corpus contributed by Nicholas Broad
- Text generated with Llama-70b and Falcon180b by Nicholas Broad
- Text generated with ChatGPT by Radek
- Official train essays
- Essays I generated with various LLMs
- EssayID if available
- Generation prompt if available
- Random 10 fold split stratified by source dataset

3.2 The Model

Create the model:

```
# We choose 512 because it's the limit of DistilBert
SEQ_LENGTH = 512

# Use a shorter sequence length.
preprocessor = keras_nlp.models.DistilBertPreprocessor.from_preset(
    "distil_bert_base_en_uncased",
    sequence_length=SEQ_LENGTH,
)

# Pretrained classifier.
classifier = keras_nlp.models.DistilBertClassifier.from_preset(
    "distil_bert_base_en_uncased",
    num_classes=2,
    activation=None,
    preprocessor=preprocessor,
)

# Re-compile (e.g., with a new learning rate)
classifier.compile(
loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
optimizer=keras.optimizers.Adam(5e-4),
    metrics=[
keras.metrics.SparseCategoricalAccuracy()
   ]
)

# Access backbone programmatically (e.g., to change `trainable`).
```

```
classifier.backbone.trainable = False


classifier.summary()
```

OUTPUT:

Preprocessor: "distil_bert_preprocessor"

Tokenizer (type)	Vocab #
distil_bert_tokenizer (DistilBertTokenizer)	30,522

Model: "distil_bert_classifier"

Layer (type)	Output Shape	Param #	Connected to
padding_mask (InputLayer)	(None, None)	0	-
token_ids (InputLayer)	(None, None)	0	-
distil_bert_backbone (DistilBertBackbone)	(None, None, 768)	66,362,880	padding_mask[0][0], token_ids[0][0]
get_item (GetItem)	(None, 768)	0	distil_bert_backbone[0][0]
pooled_dense (Dense)	(None, 768)	590,592	get_item[0][0]
output_dropout (Dropout)	(None, 768)	0	pooled_dense[0][0]
logits (Dense)	(None, 2)	1,538	output_dropout[0][0]

Total params: 66,955,010 (255.41 MB)
 Trainable params: 592,130 (2.26 MB)
 Non-trainable params: 66,362,880 (253.15 MB)

3.3 The Entire Code:

```
"""# Explore the dataset

Let's look at the distribution of labels in the training set.
"""

df_train_prompts=pd.read_csv(DATA_DIRL + "train_prompts.csv")
print(df_train_prompts.info())
df_train_prompts.head()

"""**Only two prompts are used in this dataset.**

Let's look at the distribution of text/generated in the training set.
"""

df_train_essays=pd.read_csv(DATA_DIRL + "train_essays.csv")
print(df_train_essays.info())
df_train_essays.head()

f, ax = plt.subplots(figsize=(12, 4))

sns.despine()
ax=sns.countplot(data=df_train_essays, x="prompt_id")

abs_values = df_train_essays['prompt_id'].value_counts().values

ax.bar_label(container=ax.containers[0], labels=abs_values)

ax.set_title("Distribution of prompt ID")

f, ax = plt.subplots(figsize=(12, 4))

sns.despine()
ax = sns.countplot(data=df_train_essays,
                   x="generated")

abs_values = df_train_essays['generated'].value_counts().values

ax.bar_label(container=ax.containers[0], labels=abs_values)

ax.set_title("Distribution of Generated Text")
```

"""1375 essays are written by human and only 3 by AI.****

****The distribution between the two prompts is pretty equal.****
"""

```
df_test_essays = pd.read_csv(DATA_DIRL + "test_essays.csv")
print(df_test_essays.info())
df_test_essays.head()

df_test_essays["text"].apply(lambda x : len(x))
```

"""The test dataset contains only 3 essays. The length of each essay is very small (12 characters).

 # Add new data to the training dataset

As the dataset does not contain any generated data. We will use the dataset created by [DAREK

```
KŁECZEK](https://www.kaggle.com/competitions/llm-detect-ai-generated-text/discussion/455517)
  """

  df_train_essays_ext = pd.read_csv(DATA_DIR+'/daigt-proper-train-dataset/train_drcat_04.csv')

  df_train_essays_ext.rename(columns = {"label":"generated"}, inplace=True)

  df_train_essays_ext.info()

  df_train_essays_ext.head()

  f, ax = plt.subplots(figsize=(12, 4))

  sns.despine()
  ax=sns.countplot(data=df_train_essays_ext, x="generated")

  abs_values=df_train_essays_ext['generated'].value_counts().values

  ax.bar_label(container=ax.containers[0], labels=abs_values)

  ax.set_title("Distribution of Generated Text")

  df_train_essays

  df_train_essays_final=pd.concat([df_train_essays_ext[["text", "generated"]], df_train_essays[["text", "generated"]]])

  df_train_essays_final.info()

  """# Prepare data

  Let's count the number of words in each essay
  """

  df_train_essays["text_length"]=df_train_essays["text"].apply(lambda x : len(x.split()))

  fig = plt.figure(figsize=(40,50))
  plot=sns.displot(data=df_train_essays, x="text_length", bins=30, kde=True)
  plot.fig.suptitle("Distribution of the length per essay - Train dataset")

  df_train_essays["text_length"].mean() + df_train_essays["text_length"].std()

  """# Create the model"""

  # We choose 512 because it's the limit of DistilBert
  SEQ_LENGTH = 512

  # Use a shorter sequence length.
  preprocessor = keras_nlp.models.DistilBertPreprocessor.from_preset(
      "distil_bert_base_en_uncased",
      sequence_length=SEQ_LENGTH,
  )

  # Pretrained classifier.
  classifier = keras_nlp.models.DistilBertClassifier.from_preset(
      "distil_bert_base_en_uncased",
      num_classes=2,
      activation=None,
      preprocessor=preprocessor,
  )

  # Re-compile (e.g., with a new learning rate)
  classifier.compile(

loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
      # optimizer=keras.optimizers.Adam(0.0005)
    #   metrics=[
    # keras.metrics.SparseCategoricalAccuracy()
    #  ]
  )


  # Access backbone programmatically (e.g., to change `trainable`).
  classifier.backbone.trainable = False


  classifier.summary()

  # Split the dataset into train and test sets
```

```
  from sklearn.model_selection import
train_test_split
  X_train, X_test, y_train, y_test =
train_test_split(df_train_essays_fin
al["text"],

df_train_essays_final["generated"],

test_size=0.33,

random_state=42)

  # Fit
  classifier.fit(x=X_train,
                 y=y_train,

validation_data=(X_test, y_test),
                 epochs=1,
                 batch_size=64
                )

  def displayConfusionMatrix(y_true,
y_pred, dataset):
      disp =
ConfusionMatrixDisplay.from_predicti
ons(
          y_true,
          np.argmax(y_pred, axis=1),
          display_labels=["Not
Generated","Generated"],
          cmap=plt.cm.Blues
      )

      tn, fp, fn, tp =
confusion_matrix(y_true,
np.argmax(y_pred, axis=1)).ravel()
      f1_score = tp / (tp+((fn+fp)/2))

      disp.ax_.set_title("Confusion
Matrix on " + dataset + " Dataset --
F1 Score: " + str(f1_score.round(2)))

  y_pred_test =
classifier.predict(X_test)

  from sklearn.metrics import
ConfusionMatrixDisplay,
confusion_matrix
  displayConfusionMatrix(y_test,
y_pred_test, "Test")
```

4 Results

Our model achieves promising results in distinguishing between AI-generated and human-authored text. Evaluation on a held-out test set demonstrates robust performance, with high accuracy in classification. Confusion matrix analysis reveals the model's ability to correctly identify the majority of AI-generated and human-authored essays.

4.1 Comparison to Baselines:

We compare our model against baseline approaches, including traditional machine learning classifiers and simpler deep learning architectures. Our model outperforms these baselines, highlighting the effectiveness of leveraging advanced transformer-based architectures like DistilBERT for text classification tasks.

4.2 Analysis and Discussion:

The success of our model underscores the importance of utilizing pre-trained transformer architectures for challenging NLP tasks. By fine-tuning DistilBERT on the essay dataset, our model learns to capture intricate patterns in the text data, enabling accurate classification. Additionally, the model's performance indicates its potential utility in real-world applications for detecting AI-generated content.

5 Conclusion

Finally, we present a model based on the DistilBERT architecture for detecting AI-generated text. The model demonstrates strong performance in classifying essays as either AI-generated or human-authored, showcasing the effectiveness of leveraging transformer-based models for NLP tasks. Future work may involve exploring larger datasets and fine-tuning strategies to further improve the model's performance. Our model is not more accurate than present day AI text detectors like ZeroGPT or similar tools but it’s not either less efficient than any of existing tools.

References

[1] S. R. Bowman, L. Vilnis, O. Vinyals, et al., "Generating Sentences from a Continuous Space," arXiv:1511.06349, 2015.

[2] V. Sanh, L. Debut, J. Chaumond, et al., "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," arXiv:1910.01108, 2019

[3] A. Vaswani, N. Shazeer, N. Parmar, et al., "Attention Is All You Need," arXiv:1706.03762, 2017.

[4] J. Devlin, M. Chang, K. Lee, et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," arXiv:1810.04805, 2018.

[5] D. Klęczek, "Detect AI Generate Text: LLNLP 2022 Competition," Kaggle, [Online]. Available: https://www.kaggle.com/competitions/llm-detect-ai-generated-text/discussion/455517.

[6] A. Esteva, A. Robicquet, B. Ramsundar, et al., "A guide to deep learning in healthcare," Nature Medicine, vol. 25, no. 1, pp. 24-29, 2019.

[7] C. D. Manning, P. Raghavan, H. Schütze, "Introduction to Information Retrieval," Cambridge University Press, 2008.