Classifying human written text from GPT-2 generated text

Group 3: Millie, Komal and Zoe

Introduction

- **Primary task**
  - Binary classification to differentiate fake text from human-generated text.
  - Models - Logistic Regression, Convolutional Neural Network and BERT.
- **Background**
  - GPT-2 models (117M, 345M, 762M and 1542M) released in stages because weaponizing language models is a serious concern.
- **Project motivation**
  - Contribute to the field of fake text detection by building classification models designed for the primary task.

Previous Works

Understanding Convolutional Neural Networks for Text Classification (2018)

a. Talks about the challenges involved with using CNN for text classification since unlike images text is discrete data.
b. Authors demonstrated binary classification task for sentiment analysis

Factuality Classification Using the Pre-trained Language Representation Model BERT (2019)

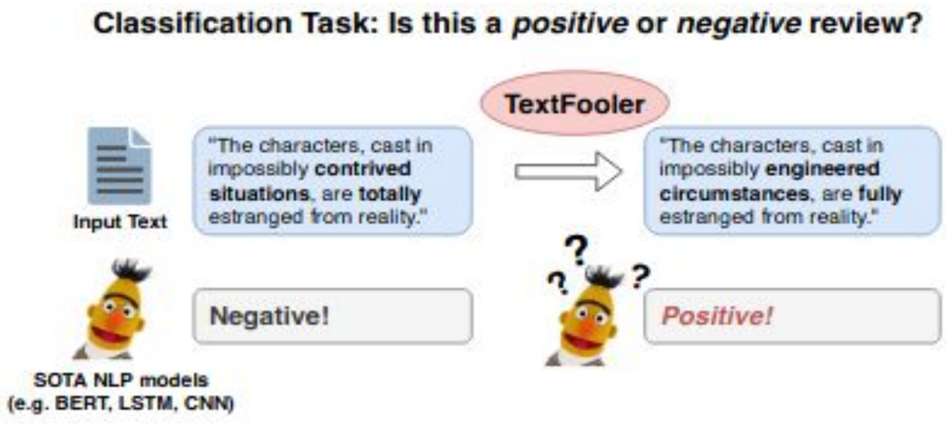
a. Factuality detection : task of assigning factual tag to verbal events present in a dataset
b. Used multi-layer bidirectional BERT model to showcase multilingual fact classification.

Previous Works

DocBERT: BERT for Document Classification (2019)

a. Used logistic regression and SVM (both inherently discriminative) for baselines to highlight the effectiveness of BERT.
b. showcased that even though BERT is generative it can give high performance in classification tasks.

Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment (2020)



Data

- Human generated text consists of 250K documents from the WebText dataset, which is a collection of text from blogs and papers.
- The GPT-2 generated text includes two 250K documents, labeled Temperature-1 and Top-k40 (hyperparameters of the GPT-2 model that affect the randomness in output).
- Top-K 40: Conditionally generated samples from the paper use top-k random sampling with k = 40.
- Sampled the first 50K human generated texts and the first 50K GPT-2 generated text of Top-K 40 configuration.

Summary Statistics on Dataset

Summary statistics for 100k dataset

	100k dataset	50k GPT-2	50k Webtext
average words/document	560	522	597
maximum words			4301
minimum words			2
average sentences/documents			23
maximum sentences			257
minimum sentences	1	1	1
vocabulary size	296067	NA	NA

	0	1
Train	30118	29882
Validation	9876	10124
Test	10006	9994

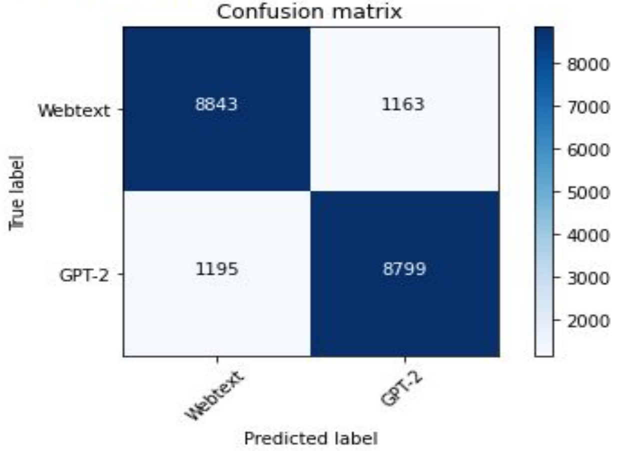
Model, Results and Analysis

- Computing infrastructure: GPU from Google Colab
- Logistic Regression
- Convolutional Neural Network
- BERT

Logistic Regression

- F1 score : 88 %
- Marginally better at predicting GPT2 as compared to WebText

```
Confusion matrix, without normalization
[[8843 1163]
 [1195 8799]]
```



	precision	recall	f1-score	support
Webtext	0.88	0.88	0.88	10006
GPT-2	0.88	0.88	0.88	9994
accuracy			0.88	20000
macro avg	0.88	0.88	0.88	20000
weighted avg	0.88	0.88	0.88	20000

Logistic Regression Analysis

Test string 1 (GPT2) : "James Harden is shooting 29 percent down the floor when guarded by Kawhi Leonard in this series, including just 10 percent from three. When not guarded by Leonard, Harden is shooting nearly 50 percent."

```
array([[0.11316767, 0.88683233]])
```

Test String 2 (WebText) : "COPYRIGHT NOTICE

This project is licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported License.

Feel free to use, copy, modify, publish and distribute, including commercial products or services."

```
array([[0.97010162, 0.02989838]])
```

CNN - engineering

- Code source - adapted from the CNN tutorial
- Tokenizer - spacy for english
- Model -

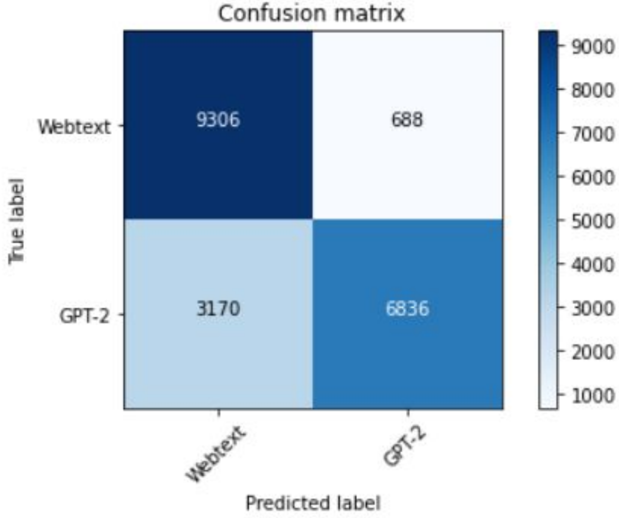
```
CNN_Text(
  (embeddings): Embedding(146142, 300)
  (convolution_layers): ModuleList(
    (0): Conv2d(1, 32, kernel_size=(2, 300), stride=(1, 1))
    (1): Conv2d(1, 32, kernel_size=(3, 300), stride=(1, 1))
    (2): Conv2d(1, 32, kernel_size=(4, 300), stride=(1, 1))
  )
  (dropout): Dropout(p=0.5, inplace=False)
  (fc): Linear(in_features=96, out_features=2, bias=True)
)
```

- Optimizer is Adam with a learning rate of 0.0001
- Loss function is CrossEntropyLoss

CNN - results

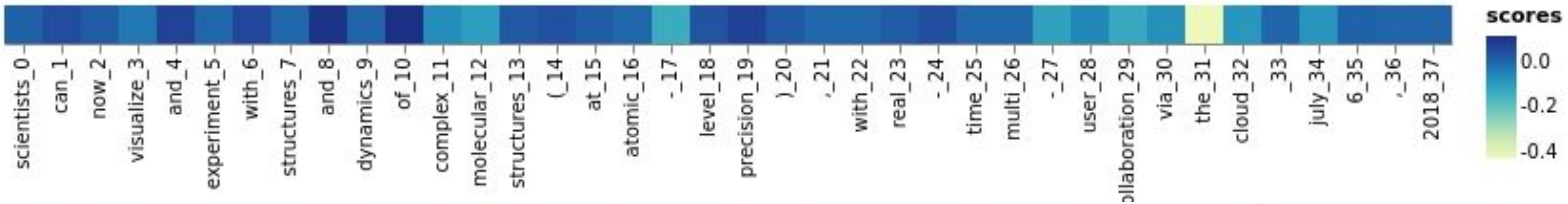
- F-score: 80.41%
- Model had difficulty correctly identifying GPT-2 text

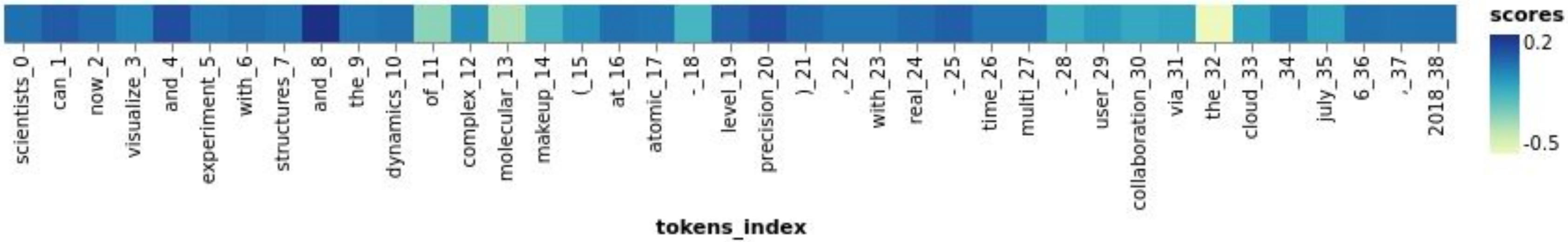
```
Overall f-score: 0.8041
Classification report:
              precision    recall  f1-score   support

     Webtext       0.75      0.93      0.83      9994
       GPT-2       0.91      0.68      0.78     10006

    accuracy                           0.81     20000
   macro avg       0.83      0.81      0.80     20000
weighted avg       0.83      0.81      0.80     20000

Confusion matrix, without normalization
[[9306  688]
 [3170 6836]]
```



CNN - analysis

	word	score		word	score
1	pendleton	0.74		cooke	-12.57
2	sandwiched	0.72		unsigned	-4.37
3	c'mon	0.69		prev	-1.52
4	leland	0.67		mick	-1.1
5	yeager	0.64		laird	-1.01
6	intermediate	0.62		mel	-0.89
7	weaves	0.61		capito	-0.82
8	aseptic	0.61		536	-0.79
9	steered	0.61		bsi	-0.71
10	straightening	0.6		pepe	-0.7

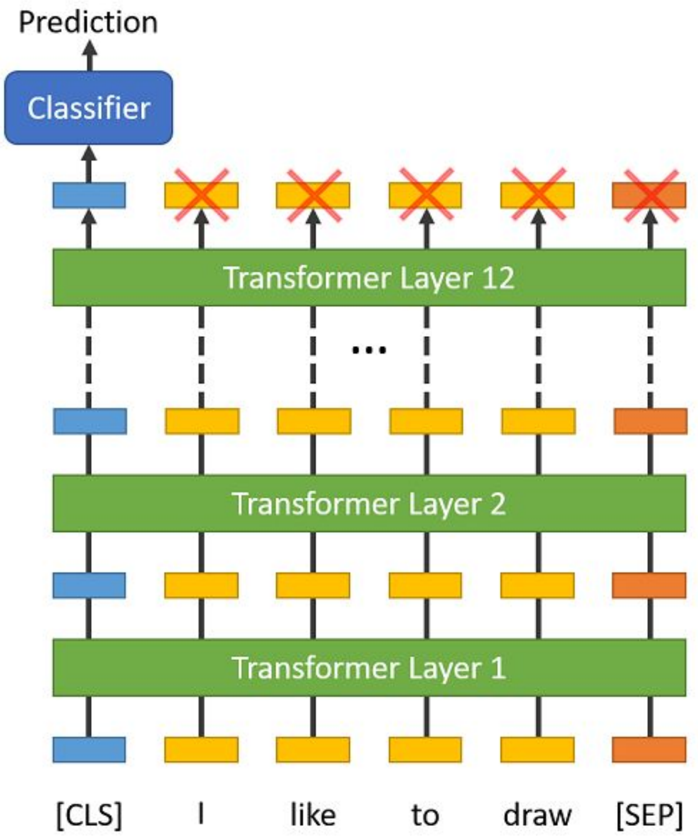
- **Findings:** sequence length and repetition are important features.
- 

- 

-

BERT - engineering

- Code source - BertForSequenceClassification from the huggingface pytorch implementation
- Tokenizer - BertTokenizer included with BERT (bert-based-uncased version)
- On the output of the final (12th) transformer, only the first embedding (the [CLS] token) is used by the classifier.



BERT - engineering

- Model -

```
BertForSequenceClassification(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
```

```
(pooler): BertPooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()
    )
  )
  (dropout): Dropout(p=0.1, inplace=False)
  (classifier): Linear(in_features=768, out_features=2, bias=True)
)
```

- Optimizer is AdamW with learning rate = 2e-5 and epsilon = 1e-8

BERT - experiment on MAX_LEN

- To examine how the truncating affect BERT's performance, we tried to set the MAX_LEN = 32, 64, 128.
- Our model performed consistently well across all sequence lengths.
- Training BERT on longer sentences made the model more prone to overfitting.

BERT performance by sequence length:

Max length	Precision		Recall		F1 score	
	Webtext	GPT-2	Webtext	GPT-2	Webtext	GPT-2
32	0.85	0.97	0.97	0.83	0.91	0.89
64	0.89	0.98	0.98	0.88	0.93	0.92
128	0.85	0.99	0.99	0.82	0.91	0.90

BERT - results

- F-score: 90.33%
- Model had difficulty correctly identifying GPT-2 text
- Error rate = 9.6%

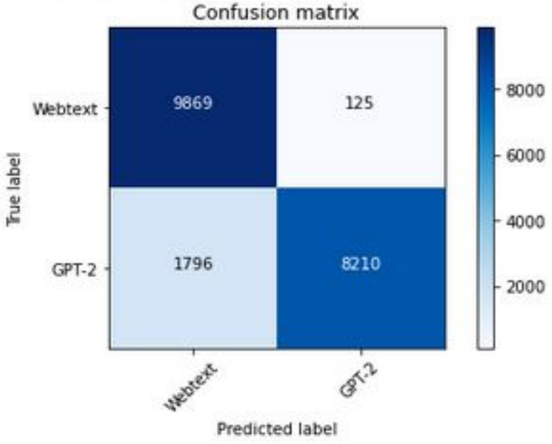
Macro F1 score: 0.9032845296211757

Classification report:

	precision	recall	f1-score	support
GPT-2	0.99	0.82	0.90	10006
Webtext	0.85	0.99	0.91	9994
accuracy			0.90	20000
macro avg	0.92	0.90	0.90	20000
weighted avg	0.92	0.90	0.90	20000

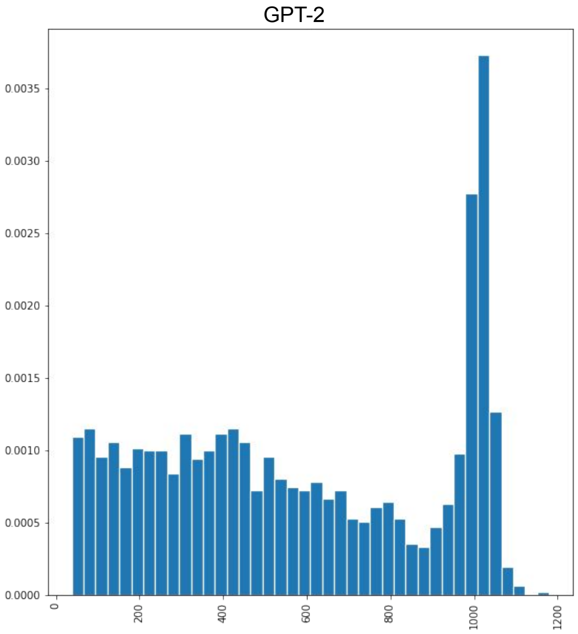
Confusion matrix, without normalization

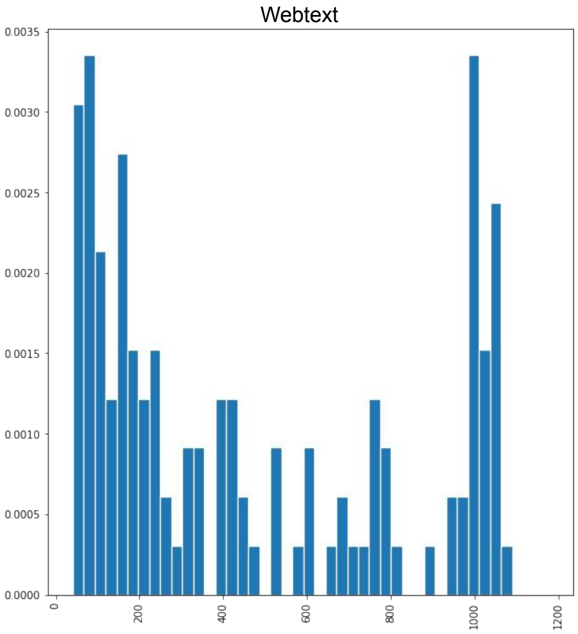
```
[[9869  125]
 [1796 8210]]
```



BERT - analysis

- The shorter Webtext documents were more prone to wrong predictions than GPT-2 data.





- At around 1000 tokens, there is a spike in wrong predictions for both sources, showing that truncating documents adversely affect BERT's performance.

Conclusion

- BERT is better but at a high training cost.

Model	Epochs	Training Time	F-score on Test Set
LR	1	< 1 minute	88.23%
CNN	20	11 minutes	80.41%
BERT	4	1.5 hour	90.34%

Future Direction

- More models to try!
  - fastText
  - GLTR
  - GPT-2
  - RoBERTa
  - etc!

References

Understanding Convolutional Neural Networks for Text Classification	https://www.aclweb.org/anthology/W18-5408/
Factuality Classification Using the Pre-trained Language Representation Model BERT (2019)	http://ceur-ws.org/Vol-2421/FACT_paper_3.pdf
DocBERT: BERT for Document Classification (2019)	https://arxiv.org/abs/1904.08398
Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment (2020) Image on slide 3	https://arxiv.org/abs/1907.11932
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (2019) Image on slide 13	https://arxiv.org/pdf/1810.04805.pdf