a creative take on AI-generated text detection:

Reverse Transformers

By Zoe Matrullo and Kilian Schulz

After going through an infinite amount of Kaggle competitions to draw inspiration from to start our project, we chose a risky one.
AI generated text recognition is a difficult task, in front of which even the pioneers of Artificial Intelligence such as OpenAI struggle, discontinuing their own tool due to inaccuracy during summer of 2023.
Nonetheless intrigued by the challenge, we decided to experiment and tried to come up with a *creative solution* that would yield positive results.

There were two possible routes for us to take:
The first was training a classifier based on large amounts of text, AI generated and human, and then let the AI do the work.
While this is definitely not easy, it is probably the computer science equivalent of choosing a recipe from www.taste.com and then following the step by step instructions. We wanted to cook our own meal, based on the principles we learned in class and come up with a more creative second option.
We ended up calling our model ***Reverse Transformer***, not because it is actually a Reverse Transformer bit by bit, but because the approach we took for figuring out if text was written by a transformer model was the result of an attempt to reverse engineer a transformer.

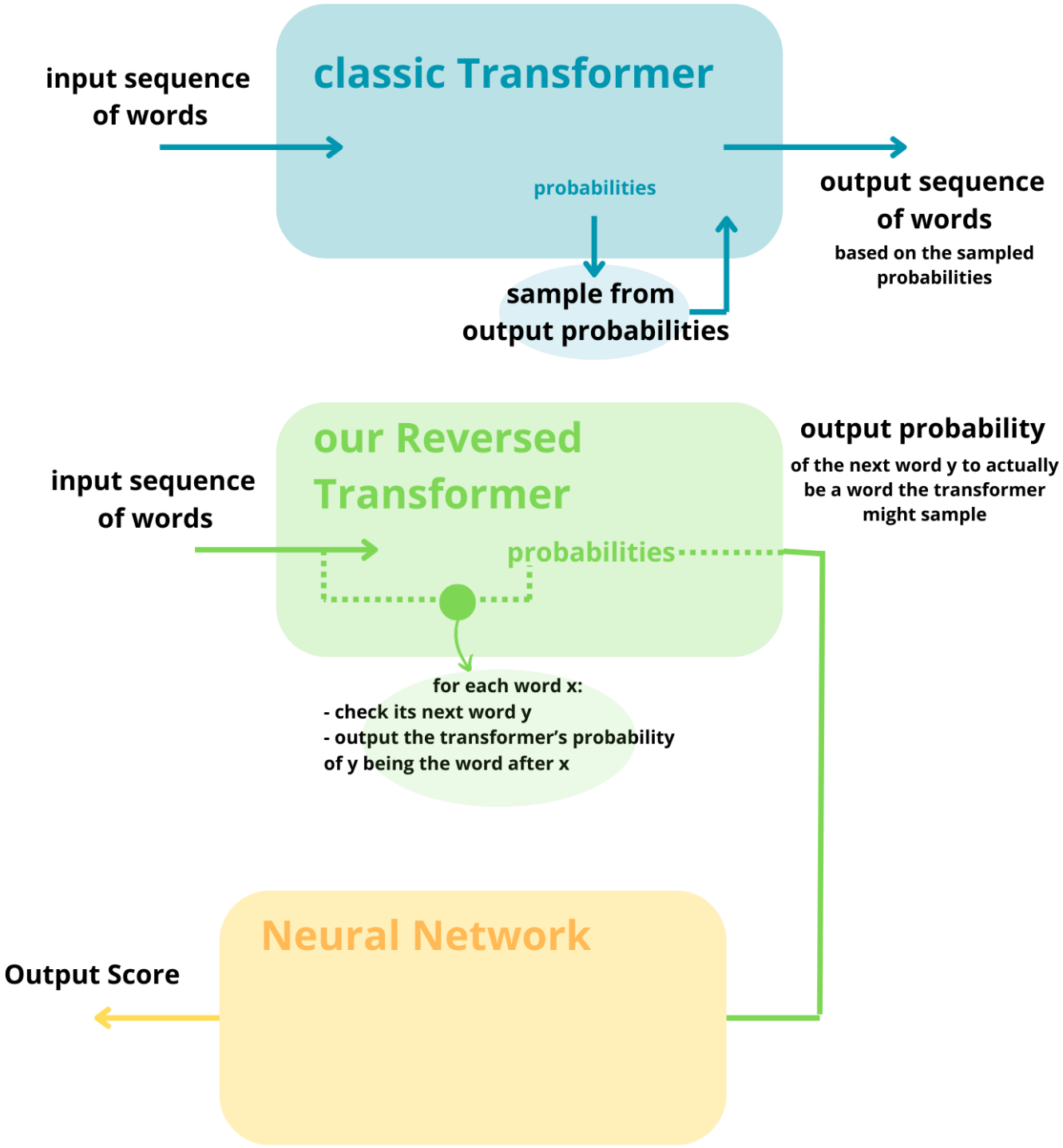
Our model

As we learned in class, the classic transformer processes an input list of words thanks to the attention mechanism and a series of feed forward neural networks, and then samples the next word according to the given output probabilities.

Our model aims at recognising AI generated text by exploiting the mentioned architecture of but with a twist:
We have a **transformer** that takes
- as **input** the sentence so far
- ***outputs*** *for each pair of words of the sentence, i.e.* ***word x*** *and* ***word y,*** *where* ***word y comes after word x in the input sentence*** *the actual probability that word y was going to be after word x according to the transformers' usual computations.*

If we have a certain degree of unpredictability, of less likely words that come up and twist in a sentence, we say that the test is likely human. If all those probabilities seem like they might have been sampled according to a probability distribution from an AI, we say that the text is likely AI.

Disclaimer: We know that it is barely possible to train an AI that recognizes state of the art generated text with a high degree of precision, so in this project we focused on the gpt-2 model to generate the training and test datasets. As we already mentioned, OpenAi themselves took down their model that was supposed to recognise AI generated text, since nowadays GPT4, Bard and others simply have such a complicated and advanced structure, that it is impossible or very inaccurate to distinguish from human written text.


input sequence of words
classic Transformer
probabilities
output sequence of words
based on the sampled probabilities
sample from output probabilities
input sequence of words
our Reversed Transformer
probabilities
output probability
of the next word y to actually be a word the transformer might sample
for each word x:
- check its next word y
- output the transformer's probability of y being the word after x
Neural Network
Output Score

The code

The project has **3 main components:**

- **the dataset**
- **the Reverse Transformer**
- **the Feed Forward Neural Network**

Let's delve deeper into each of them.

the dataset

(please refer to the /data/ directory of our GitHub repository)

To start an experiment and prove if our intuition could lead to some concrete results, we started by generating a dataset of sentences generated by both sources.

Generate AI sentences

The file **/data/create_data.py** contains the code used to create a dataset of almost 400 gpt-2 generated sentences.

We made use of the gpt-2 model kindly offered on Hugging Face's repository on GitHub linked here.

Starting from a series of random prompts, we created the sentences that are stored in data/AI_sentences.txt.

The file **/data/get_human_sentences.py** contains the functions used to get the portion of dataset made up by human sentences. These sentences were taken from songs, books and poetry scraped from the Gutenberg dataset.

To first confirm if the initial idea would work, we ran the file containing an initial draft of the Reverse Transformer, **/old_testing.py,** on both files of sentences**.**

This first test gave us promising results.

the Reverse Transformer

(please refer to the main directory of our GitHub repository)

From perfecting the first draft, we got a new version of the Reverse Transformer in **/zoes_testing.py**

This file exploits Pytorch and the gpt-2 model and is made up of 2 functions:

- ***get_next_word_probability(sentence, next_word)***
  - **inputs:**
    - ***next_word***, a word of ***sentence***
    - ***sentence***, the sentence up to the word before next_word it positionally encodes the sentence
  - creates an ***LMHead model***, a type of transformer optimized for NLP
  - runs the transformer on the sentence and gets the output probabilities
  - positionally encodes next_word
  - takes the output probability of the word corresponding to next_word
- ***process_sentences(input_file, output_files)***
  - takes the sentences from the input file
  - for each word in the sentence and a part of the sentence it runs *get_next_word_probability(sentence, next_word) and appends the results for each sentence (a list) to a bigger list, creating a list of lists.*
  - *each line of the output file will have the following structure "sentence", "probability of each word to be the following of its successor", "avg of the probabilities of each word in the sentence"*

When testing the averages obtained that:

- the AI average of probability is 0.23820649143351932
- the human average of probability is 0.09411602139188101

the Feed Forward Neural Network

now that we obtained some data on which to distinguish the two types of sentences, there were different ways of performing the classification task:

- either we could have naively chosen a threshold between the two labels, but there were obviously some outliers, just like there were in the training set, causing lost accuracy,
- using statistics computations on the probabilities, calculating how likely it was for a word to be sampled based on the output probabilities of each word,
- using a Neural Network to recognize the patterns of predictability or unpredictability of a sentence in order to classify the sentence.

We decided to go with this last option, to experiment even more with deep learning and check how accurate a Feed Forward Logistic Neural Network would be.
Our neural network has as:

- ***input*** *the list of probabilities related to each word of a sentence*
- ***output*** *a value between 0 and 1, the higher this value is the more likely the sentence is to be AI generated*

It consists of a hidden layer with ReLu activation function and two other hidden layers with sigmoid activation function.
The reason for choosing these layers is trying to keep it simple and it being the most successful version of the ones we experimented with.

Results

Since our NN is quite lightweight in order to not slow down or overfit on the training data, we only run it for 20 epochs to train it. After 20 epochs we test with around 50 samples for each class and consistently get precision scores in the low- to mid-nineties.

```
Epoch 1, Loss: 0.6187880542129278
Epoch 2, Loss: 0.36857301311101764
Epoch 3, Loss: 0.22254648082889616
Epoch 4, Loss: 0.18870082183275372
Epoch 5, Loss: 0.1688347215531394
Epoch 6, Loss: 0.15557983721373603
Epoch 7, Loss: 0.1427900663111359
Epoch 8, Loss: 0.13029344400274567
Epoch 9, Loss: 0.119308107503457
Epoch 10, Loss: 0.11105338613560889
Epoch 11, Loss: 0.09895094526291359
Epoch 12, Loss: 0.08950524451211095
Epoch 13, Loss: 0.08025978726072935
Epoch 14, Loss: 0.07266943231661571
Epoch 15, Loss: 0.06366005458039581
Epoch 16, Loss: 0.05506352206430165
Epoch 17, Loss: 0.05053733194654342
Epoch 18, Loss: 0.04454973598512879
Epoch 19, Loss: 0.038830952342323144
Epoch 20, Loss: 0.03512401292027789
Precision: 0.9230769230769231
```

Obviously this has to be taken with a grain of salt, since our Transformer is based on a GPT2 model and also the generated sentences are base on a GPT2 model, but nevertheless this shows that once you manage to create a transformer with a similar nature to the one generating the text, it is very much possible to use our approach to accurately recognize text written by an AI.

Conclusions

We have shown in this project that predictability according to a personal transformer can be a useful metric for seeing if another text has been generated by a similar transformer. In the future, this might point towards using a similar approach as an additional way of figuring out if a text is handwritten or AI generated, which is one of the big challenges the next decade might pose.