Advanced AI Project 4

Jordan White

December 3, 2021

1 Introduction

In this project, I made use of pre-trained transformer models to perform three different tasks:

1. Text generation from multiple 1, 5, and 10 word prompts.
2. Sentiment analysis of user reviews of products from a selected category sold on Amazon.
3. Predicting missing words (masked language modeling).

Credit to Hugging Face Co. for Transformer code.

Contents

2 Part One: Text Generation

2.1 Program Overview

In this portion of the project, I used a transformer that used GPT-2 to generate a sensible continuation of three prompts of lengths one, five, and ten words. For each prompt, I generated continuations containing five additional words, 20 additional words, and 100 total words. The specific text-generator I employed was GPT-2.

The one-word prompt I supplied was 'Penguins'. I wanted to see what the program could do with this oddly specific word.

The five word prompt I gave was "Why are you such a". I was interested in how GPT-2 would handle an interpersonal prompt, such as an accusation/insult.

The 10-word prompt was "When I grew up, we did not have all these". I chose this because it is somewhat of a cliche saying among older generations, and I wanted to see if the model was advanced enough to pick up on the stereotype.

2.2 Results

The passages the program came up with are shown in Figure 1.

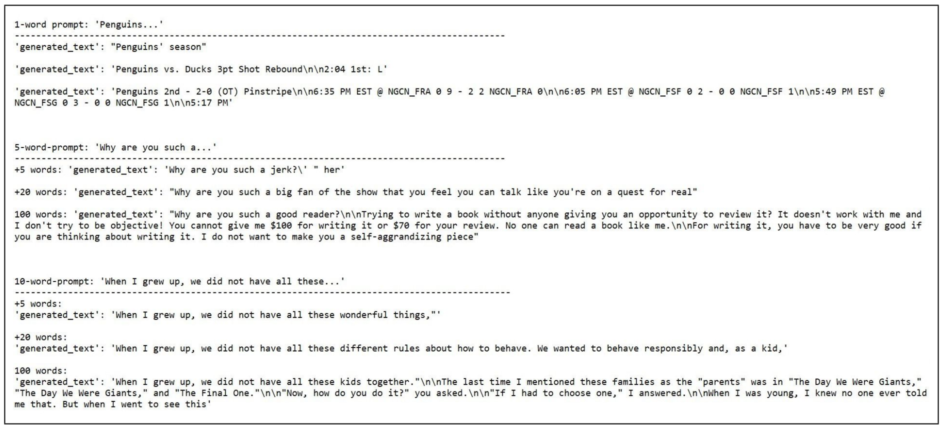
When I supplied the 1-word prompt with a generic word, like 'Well' or 'I', the program had no trouble writing something that made sense. But it had more difficulty with random first words, such as 'Penguin'. All the passages in came up with in response to this were either sports related (I suppose there are teams out there called the "Penguins"), or video game related. The resulting 100-word passage is a random string of unformatted data. Presumably, the model was trained on data from various webpages, which would explain why it inserts newline characters or timestamps.

It seems to me the program writes in a way that only takes the previous few words into account when choosing the next word. In this way, the writing sounds like it makes sense on the surface, but no underlying meaning is preserved across sentences. The best example of this is the 100-word continuation of the 5-word prompt.

For the 10-word prompt, the program was utterly unable to create sentences with higher order meaning, such as with stereotypes. It does seem able to write about a specific theme, such as the theme of 'behaviour' in the 20-word continuation of the 10-word prompt. However, the program seems willing to abandon a theme after it inserts a newline (\n) character, which is interesting behavior in its own right and something I would be interested in studying more. Perhaps the program has recognized the pattern of shifting topics after the start of a new paragraph.

All in all, I was a little disappointed with GPT-2's performance, but there was still a lot of interesting take-aways from my it. GPT-3, on the other hand, is terrifying, and makes me value these last precious few days of freedom we have before the terminators take over.

2.3 Figures



```
1-word prompt: 'Penguins...'
--------------------------------------------------------------------------------------------
'generated_text': "Penguins' season"

'generated_text': 'Penguins vs. Ducks 3pt Shot Rebound\n\n2:04 1st: L'

'generated_text': 'Penguins 2nd - 2-0 (OT) Pinstripe\n\n6:35 PM EST @ NGCN_FRA 0 9 - 2 2 NGCN_FRA 0\n\n6:05 PM EST @ NGCN_FSF 0 2 - 0 0 NGCN_FSF 1\n\n5:49 PM EST @
NGCN_FSG 0 3 - 0 0 NGCN_FSG 1\n\n5:17 PM'


5-word-prompt: 'Why are you such a...'
--------------------------------------------------------------------------------------------
+5 words: 'generated_text': 'Why are you such a jerk?\' " her'

+20 words: 'generated_text': "Why are you such a big fan of the show that you feel you can talk like you're on a quest for real"

100 words: 'generated_text': "Why are you such a good reader?\n\nTrying to write a book without anyone giving you an opportunity to review it? It doesn't work with me and
I don't try to be objective! You cannot give me $100 for writing it or $70 for your review. No one can read a book like me.\n\nFor writing it, you have to be very good if
you are thinking about writing it. I do not want to make you a self-aggrandizing piece"


10-word-prompt: 'When I grew up, we did not have all these...'
---------------------------------------------------------------------------------------------
+5 words:
'generated_text': 'When I grew up, we did not have all these wonderful things,"'

+20 words:
'generated_text': 'When I grew up, we did not have all these different rules about how to behave. We wanted to behave responsibly and, as a kid,'

100 words:
'generated_text': 'When I grew up, we did not have all these kids together."\n\nThe last time I mentioned these families as the "parents" was in "The Day We Were Giants,"
"The Day We Were Giants," and "The Final One."\n\n"Now, how do you do it?" you asked.\n\n"If I had to choose one," I answered.\n\nWhen I was young, I knew no one ever told
me that. But when I went to see this'
```

Figure 1: The generated text resulting from the 1, 5, and 10 word prompts.

2.4 Possible Applications

There are many possible applications of this task. The most obvious seems to be fiction writing, but this could easily be generalized to writing poetry, philosophy, musical lyrics, or even movie/play scripts.
Personal speculation: If text generation gets advanced enough that an AI can pass the Turing test (and there is little reason to think that will not happen), the ethical implications could be immense, as behaviorist groups could begin arguing for machine rights on the grounds that the cultural gap between humans and machines will be greatly diminished.

3 Part Two: Sentiment Analysis

3.1 Program Overview

In the portion of the project, I used Hugging Face's sentiment analysis pipeline to classify Amazon reviews as positive or negative. This was a good data set to use because I could immediately verify my results by comparing the classification of a review to its rating. For the purpose of this comparison, I assumed all ratings with 3, 4, or 5 starts are positive, while ratings with 1 or 2 stars are negative.
I used the 5-core "Gift-cards" dataset, which can be found under the "'Small' subsets for experimentation" section of this webpage. This dataset has 2972 reviews.

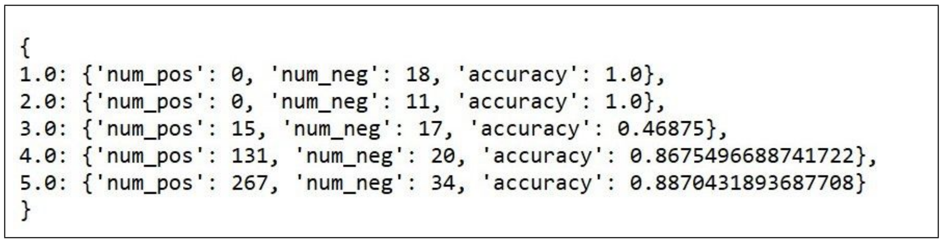
3.2 Results

I decided to analyze the performance of the algorithm on each of the five possible ratings, 1-star through 5-stars, before analyzing the programs performance holistically. What I discovered, as shown in Figure 2, is the algorithm performed perfectly (100% accuracy) for negative reviews (1 and 2 stars). This makes some sense, because we would expect 1 and 2-star reviews to use the most exaggerated and emotional language, which naturally makes them easier to classify. For 4 and 5-star reviews, the algorithm also performed well (86.7% and 88.7%, respectively). However, for 3-star reviews, the algorithm performed very poorly when operating under the "3-stars is positive" assumption: It classified these correctly only 46.8% of the time. This makes sense, because the sentiment of 3-star reviews can vary a lot since it is neither completely positive nor completely negative. I anticipated this, which is why I analyzed the performance of each rating level.

Taken all together, the program's accuracy is 86.1% over all reviews. When 3-star reviews are factored out of the equation, the program's total accuracy is 88.77%. The reason the accuracy only increased a couple percent is because my data-set only had a few dozen 1, 2, and 3-star reviews, while it had 100s of 4 and 5 star reviews. In the future I may want to acquire a larger dataset for a more accurate analysis.

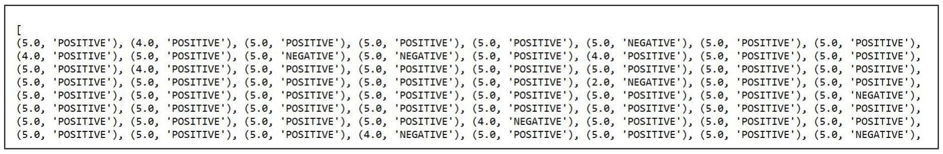
To see the data structure I used to store information about reviews, see Figure 3. In it, you can see a list of tuples. Each tuple represents one distinct review, and each tuple contains the rating (1-5 stars) of that review, as well as the prediction (positive vs negative) the program made.

3.3 Figures



```
{
1.0: {'num_pos': 0, 'num_neg': 18, 'accuracy': 1.0},
2.0: {'num_pos': 0, 'num_neg': 11, 'accuracy': 1.0},
3.0: {'num_pos': 15, 'num_neg': 17, 'accuracy': 0.46875},
4.0: {'num_pos': 131, 'num_neg': 20, 'accuracy': 0.8675496688741722},
5.0: {'num_pos': 267, 'num_neg': 34, 'accuracy': 0.8870431893687708}
}
```

Figure 2: Classification accuracy for different ratings.



```
[
(5.0, 'POSITIVE'), (4.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'NEGATIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'),
(4.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'NEGATIVE'), (5.0, 'NEGATIVE'), (5.0, 'POSITIVE'), (4.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'),
(5.0, 'POSITIVE'), (4.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'),
(5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (2.0, 'NEGATIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'),
(5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'NEGATIVE'),
(5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'),
(5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (4.0, 'NEGATIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'),
(5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (4.0, 'NEGATIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'POSITIVE'), (5.0, 'NEGATIVE'),
```

Figure 3: Main data structure containing classification and rating info for each review.

3.4 Possible Applications

The applications of this technology to marketing seem very obvious. Because of mass collection of personal data on each person, a machine may very well be able to determine the type of product that each type of person is most likely to prefer, and a machine could place people into different categories based on an emotional/sentimental analysis of what they write on the internet.

4 Part Three: Masked Language Modeling

4.1 Program Overview

In the final part of the project, I used Hugging Face's fill-mask pipeline to perform masked language modeling. This task involves analyzing a passage with one hidden word, and trying to use the context of the passage to determine the most likely words that could fill in the gap.
I used the datasets given to me, one of which contained passages with hidden nouns, another with hidden verbs, and another with hidden adjectives.
First, I analyzed the probability that the program guessed the hidden word as its first choice, then I analyzed the probability it guessed the hidden word in its top five choices.

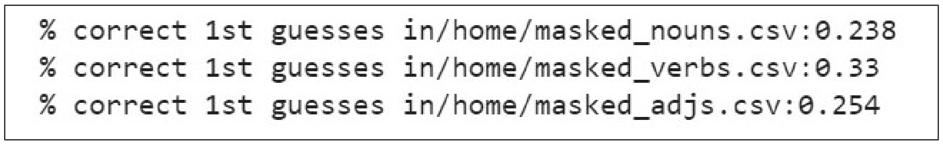
4.2 Results

As you can see in Figure 4, the probability the program guessed the word correctly on its first try was 23.8% for nouns, 33% for verbs, and 25.4% for adjectives. This averages to 27.4% accuracy total. It impresses me that it is this high, considering the tens of thousands of words in the English language.

In Figure 5, we can see the probability that the program guesses the word correctly in its top 5 choices. These probabilities are 41% for nouns, 58.2% for verbs, and 51.8% for adjectives. This average to 50.3% for all words.

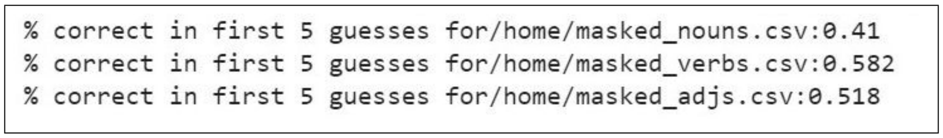
All in all, there seems to be a trend of the program performing best on verbs. I think this makes sense, because many verbs (transitive verbs in particular) are necessarily accompanied by two things, a noun subject and a noun object. Meanwhile, nouns and adjectives often have only one necessary attachment, or could be eliminated from the sentence completely. This means verbs are essential to sentences more often than nouns or adjectives, and as such there are (in general) fewer possible verbs that could fill in a word gap to make a sentence work when compared to nouns or adjectives. But I am not a linguist, so this is just my theory for why the data is this way.

4.3 Figures



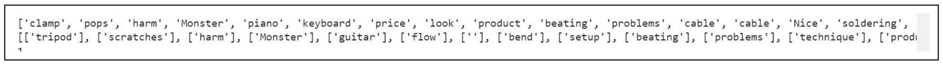
```
% correct 1st guesses in/home/masked_nouns.csv:0.238
% correct 1st guesses in/home/masked_verbs.csv:0.33
% correct 1st guesses in/home/masked_adjs.csv:0.254
```

Figure 4: Probability of a correct guess on first try.



```
% correct in first 5 guesses for/home/masked_nouns.csv:0.41
% correct in first 5 guesses for/home/masked_verbs.csv:0.582
% correct in first 5 guesses for/home/masked_adjs.csv:0.518
```

Figure 5: Probability of a correct guess in first five tries.



```
['clamp', 'pops', 'harm', 'Monster', 'piano', 'keyboard', 'price', 'look', 'product', 'beating', 'problems', 'cable', 'cable', 'Nice', 'soldering',
[['tripod'], ['scratches'], ['harm'], ['Monster'], ['guitar'], ['flow'], [''], ['bend'], ['setup'], ['beating'], ['problems'], ['technique'], ['prod
]
```

Figure 6: Top: Actual hidden words. Bottom: The word my program guessed as being most likely.

4.4 Possible Applications

The applications of Masked Language Modeling seem more subtle. It may be that MLM is used primarily for training an AI to learn the relations of words to the broader context of their sentences, for the purpose of advancing Natural Language Processing. Or MLM could be used as software for improving writing by suggesting better word choices to writers.