AUTOMATION OF PRODUCT DESCRIPTION WITH AI

PHASE 1

By: Patrick Ayers

WHAT ARE THE GOALS

Phase 1

- Is it possible to leverage the language models made by Open AI to generate product descriptions

Phase 2

- Build a data pipeline that will ensure the quality of product information.

HOW ARE WE MEASURING SUCCESS?

1. Confirm that a GPT model can produce meaningful product descriptions.
2. Confirm that what is produced is not harmful to the brand, product, or company.
3. Review the cost of running and building the model.

BACKGROUND INFORMATION

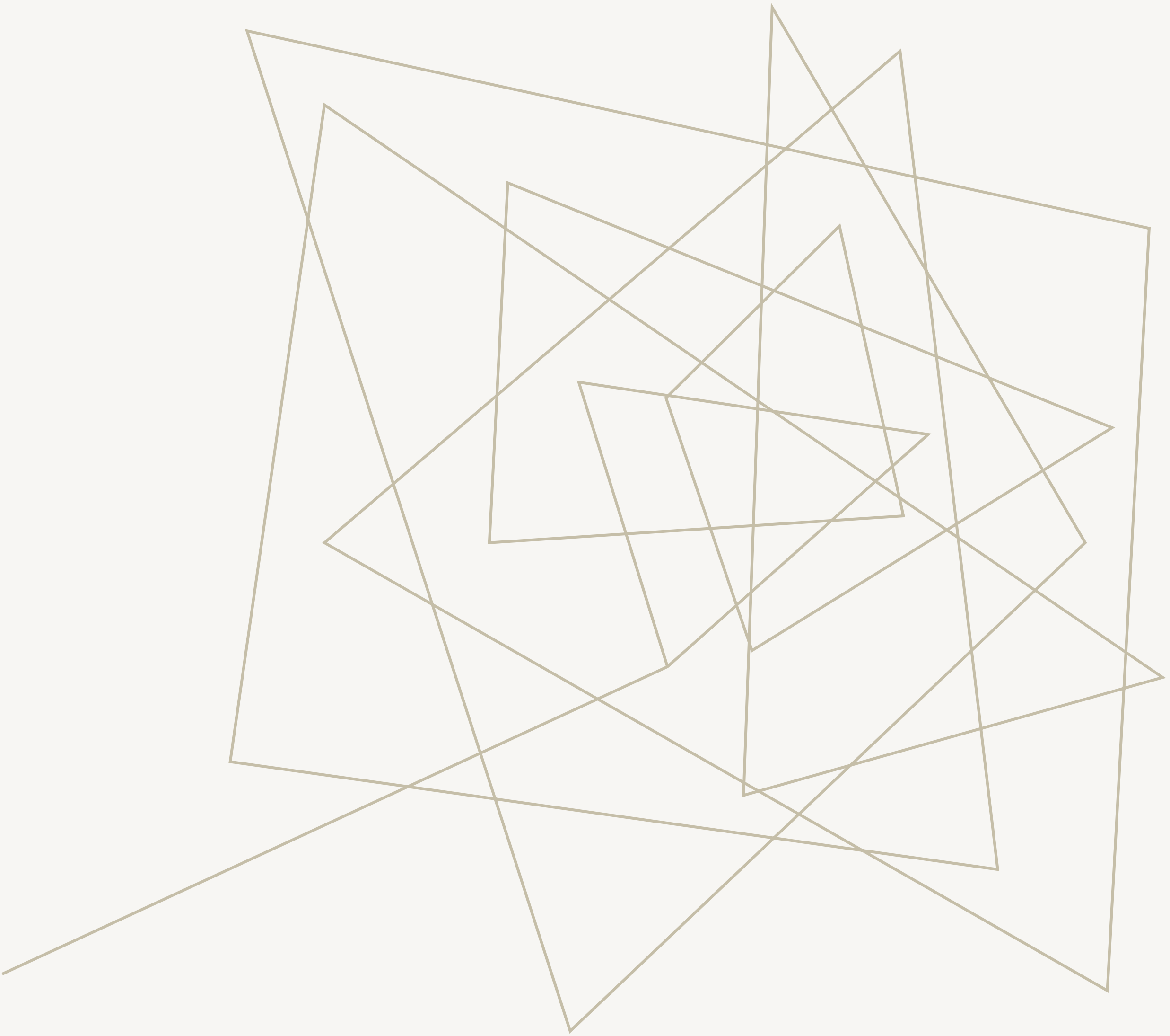
Who is Open AI?

OpenAI is a research and deployment company focusing on AI. They have made many great AI models in the last year.

- ChatGPT => Optimize language models for dialogue
- Dall-E2 => image classification and generation
- openAI codex => Natural Language to code

What model from Open AI did I use?

ChatGPT API was not available at the time of this analysis, so I needed to use a GPT-3 model called text-davinci-003.



DATA AND ANALYSIS

ABOUT THE DATA

DATA SET IS FROM TAYLOR STITCH

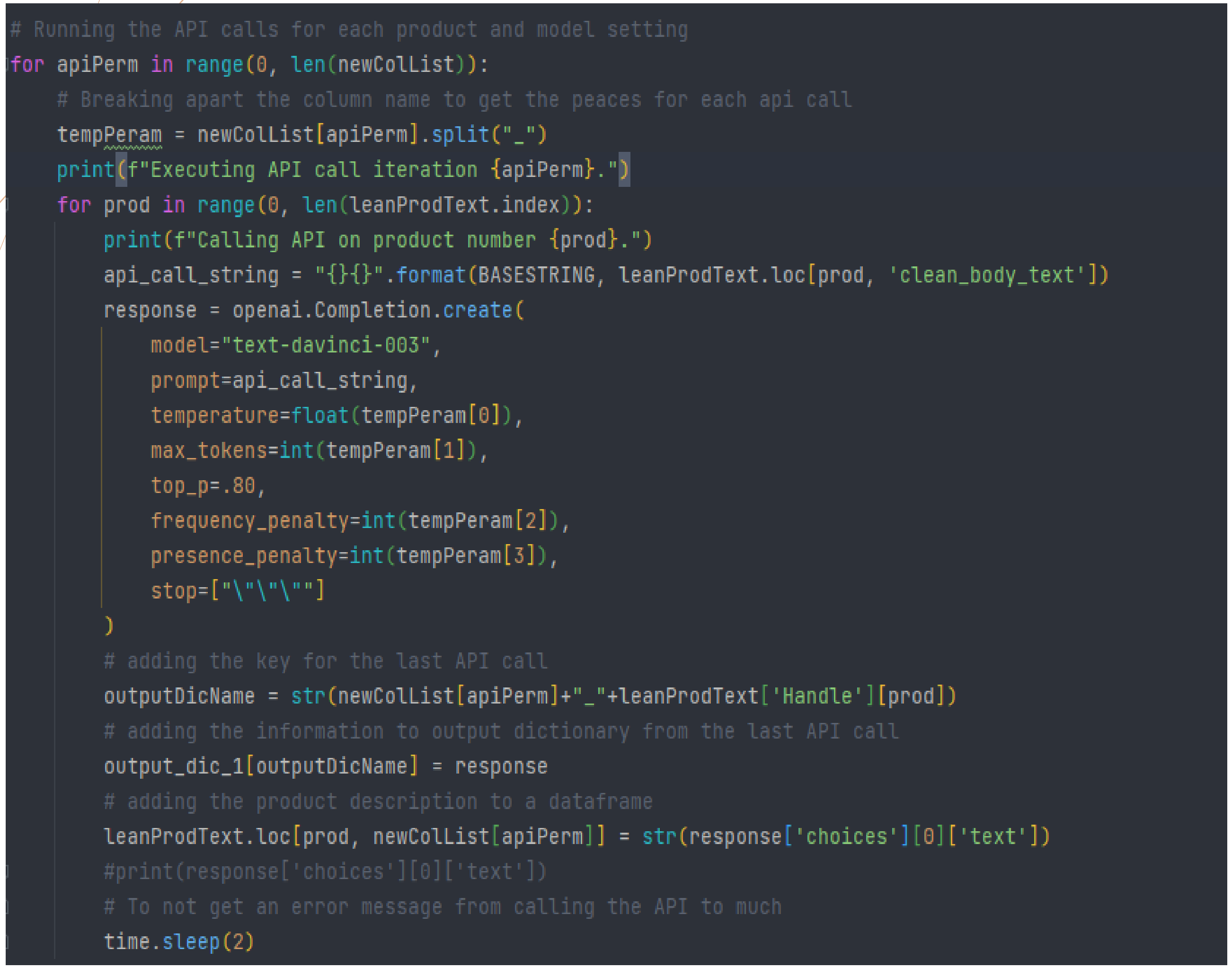
SAMPLE SIZE: 145

Handel	Title	UDA Type	UDA Vender	UDA Color
camp-sock-in-soil-marl-2309	The Camp Sock in Soil Marl	SOCK	TAYLOR STITCH	SOIL MARL
camp-sock-in-dark-navy-marl-2309	The Camp Sock in Dark Navy Marl	Sock	TAYLOR STITCH	NAVY MARL
headland-beanie-in-warm-grey-2309	The Headland Beanie in Warm Grey	BEANIE	TAYLOR STITCH	WARM GREY
headland-beanie-in-dark-navy-2309	The Headland Beanie in Dark Navy	BEANIE	TAYLOR STITCH	DARK NAVY
quilted-jersey-pant-in-midnight-heather-2309	The Quilted Jersey Pant in Midnight Heather	JERSEY	TAYLOR STITCH	MIDNIGHT HEATHER

WHY THIS DATA

- Taylor Stich uses the Shopify platform
  - Allowed me to use a Chrome add-on called Shopify scraper.
- This clothing store focuses on outdoor apparel
  - This allowed the result to be easier to analyze
- I am incredibly familiar with the apparel space

RUNNING GPT MODEL



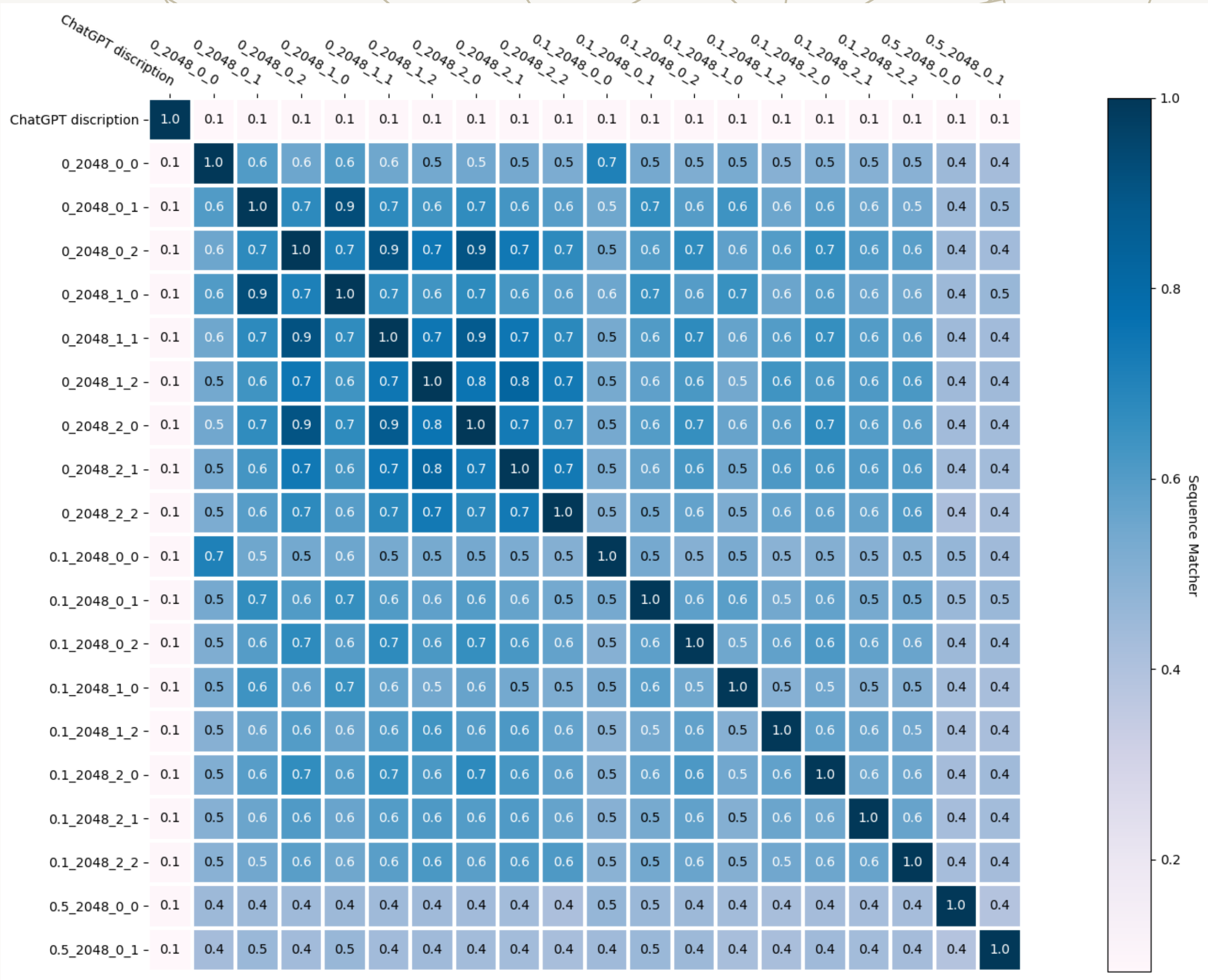
```python
# Running the API calls for each product and model setting
for apiPerm in range(0, len(newColList)):
    # Breaking apart the column name to get the peaces for each api call
    tempPeram = newColList[apiPerm].split("_")
    print(f"Executing API call iteration {apiPerm}.")
    for prod in range(0, len(leanProdText.index)):
        print(f"Calling API on product number {prod}.")
        api_call_string = "{}{}".format(BASESTRING, leanProdText.loc[prod, 'clean_body_text'])
        response = openai.Completion.create(
            model="text-davinci-003",
            prompt=api_call_string,
            temperature=float(tempPeram[0]),
            max_tokens=int(tempPeram[1]),
            top_p=.80,
            frequency_penalty=int(tempPeram[2]),
            presence_penalty=int(tempPeram[3]),
            stop=["\"\"\""]
        )
        # adding the key for the last API call
        outputDicName = str(newColList[apiPerm]+"_"+leanProdText['Handle'][prod])
        # adding the information to output dictionary from the last API call
        output_dic_1[outputDicName] = response
        # adding the product description to a dataframe
        leanProdText.loc[prod, newColList[apiPerm]] = str(response['choices'][0]['text'])
        #print(response['choices'][0]['text'])
        # To not get an error message from calling the API to much
        time.sleep(2)
```

- **Prompt**:
  - Phase used to generate completions from.
  - Rewrite this description but with a tone of great quality and fashionable
- **Temperature**:
  - The model will take more risks
- **Presency_penalty**
  - (-2,2)
  - Positive values penalize new tokens based on whether they appear in the text
- **Prequency_penalty**
  - (-2,2)
  - Positive values penalize new tokens based on their existing frequency in the text. Decreasing the model's likelihood of repeating the same line verbatim.

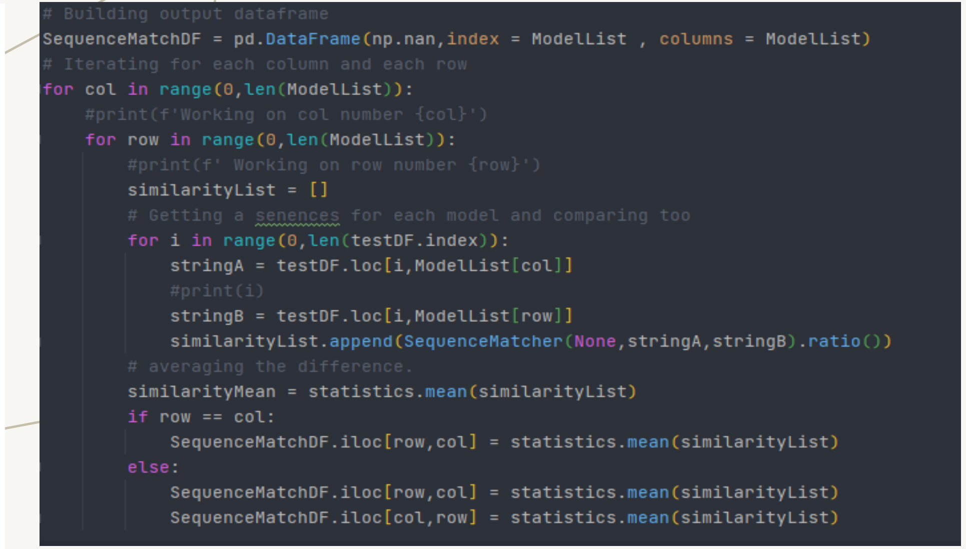
OUTPUT

Clean Text (input)	0_2048_0_1	0.5_2048_0_2	ChatGPT (Used Opening AI website)
Rewrite this description but with a tone of great quality and fashionable. The Camp Sock in Soil Marl from Taylor Stitch	The Camp Sock in Soil Marl from Taylor Stitch is an exquisite piece that will elevate your style. This luxurious sock is the perfect way to add a touch of sophistication to any ensemble.	The Camp Sock in Soil Marl from Taylor Stitch is an exquisite, luxurious piece of legwear that provides both fashion-forward style and unparalleled comfort.	Experience unparalleled luxury and style with the Camp Sock in the timeless Soil Marl colorway from the elite fashion label, Taylor Stitch. Meticulously crafted with the highest quality materials, this sock exudes sophistication and fashion-forward design, elevating any look with its refined aesthetic

COMPARING PRODUCT DESCRIPTIONS



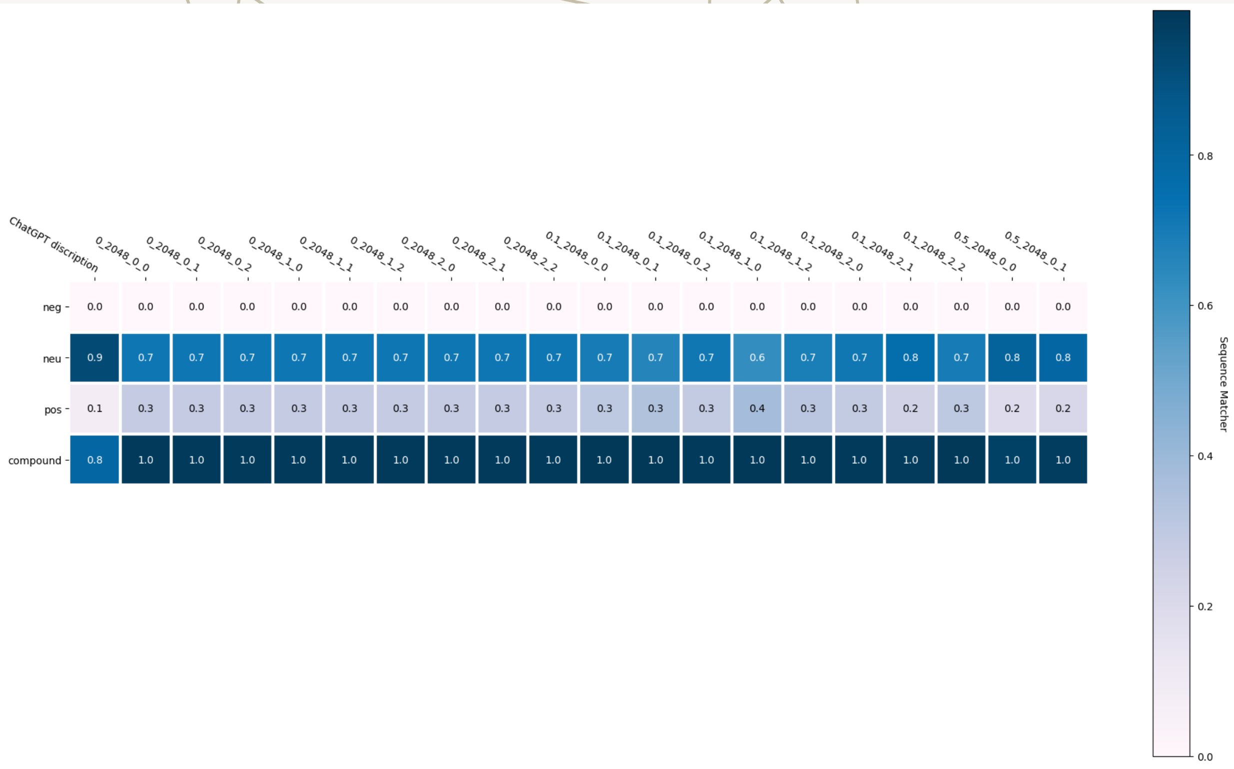
	ChatGPT discription	0_2048_0_0	0_2048_0_1	0_2048_0_2	0_2048_1_0	0_2048_1_1	0_2048_1_2	0_2048_2_0	0_2048_2_1	0_2048_2_2	0.1_2048_0_0	0.1_2048_0_1	0.1_2048_0_2	0.1_2048_1_0	0.1_2048_1_2	0.1_2048_2_0	0.1_2048_2_1	0.1_2048_2_2	0.5_2048_0_0	0.5_2048_0_1
ChatGPT discription	1.0	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0_2048_0_0	0.1	1.0	0.6	0.6	0.6	0.6	0.5	0.5	0.5	0.5	0.7	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.4	0.4
0_2048_0_1	0.1	0.6	1.0	0.7	0.9	0.7	0.6	0.7	0.6	0.6	0.5	0.7	0.6	0.6	0.6	0.6	0.6	0.5	0.4	0.5
0_2048_0_2	0.1	0.6	0.7	1.0	0.7	0.9	0.7	0.9	0.7	0.7	0.5	0.6	0.7	0.6	0.6	0.7	0.6	0.6	0.4	0.4
0_2048_1_0	0.1	0.6	0.9	0.7	1.0	0.7	0.6	0.7	0.6	0.6	0.6	0.7	0.6	0.7	0.6	0.6	0.6	0.6	0.4	0.5
0_2048_1_1	0.1	0.6	0.7	0.9	0.7	1.0	0.7	0.9	0.7	0.7	0.5	0.6	0.7	0.6	0.6	0.7	0.6	0.6	0.4	0.4
0_2048_1_2	0.1	0.5	0.6	0.7	0.6	0.7	1.0	0.8	0.8	0.7	0.5	0.6	0.6	0.5	0.6	0.6	0.6	0.6	0.4	0.4
0_2048_2_0	0.1	0.5	0.7	0.9	0.7	0.9	0.8	1.0	0.7	0.7	0.5	0.6	0.7	0.6	0.6	0.7	0.6	0.6	0.4	0.4
0_2048_2_1	0.1	0.5	0.6	0.7	0.6	0.7	0.8	0.7	1.0	0.7	0.5	0.6	0.6	0.5	0.6	0.6	0.6	0.6	0.4	0.4
0_2048_2_2	0.1	0.5	0.6	0.7	0.6	0.7	0.7	0.7	0.7	1.0	0.5	0.5	0.6	0.5	0.6	0.6	0.6	0.6	0.4	0.4
0.1_2048_0_0	0.1	0.7	0.5	0.5	0.6	0.5	0.5	0.5	0.5	0.5	1.0	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.4
0.1_2048_0_1	0.1	0.5	0.7	0.6	0.7	0.6	0.6	0.6	0.6	0.5	0.5	1.0	0.6	0.6	0.5	0.6	0.5	0.5	0.5	0.5
0.1_2048_0_2	0.1	0.5	0.6	0.7	0.6	0.7	0.6	0.7	0.6	0.6	0.5	0.6	1.0	0.5	0.6	0.6	0.6	0.6	0.4	0.4
0.1_2048_1_0	0.1	0.5	0.6	0.6	0.7	0.6	0.5	0.6	0.5	0.5	0.5	0.6	0.5	1.0	0.5	0.5	0.5	0.5	0.4	0.4
0.1_2048_1_2	0.1	0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.5	0.5	0.6	0.5	1.0	0.6	0.6	0.5	0.4	0.4
0.1_2048_2_0	0.1	0.5	0.6	0.7	0.6	0.7	0.6	0.7	0.6	0.6	0.5	0.6	0.6	0.5	0.6	1.0	0.6	0.6	0.4	0.4
0.1_2048_2_1	0.1	0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.5	0.5	0.6	0.5	0.6	0.6	1.0	0.6	0.4	0.4
0.1_2048_2_2	0.1	0.5	0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.5	0.5	0.6	0.5	0.5	0.6	0.6	1.0	0.4	0.4
0.5_2048_0_0	0.1	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.5	0.5	0.4	0.4	0.4	0.4	0.4	0.4	1.0	0.4
0.5_2048_0_1	0.1	0.4	0.5	0.4	0.5	0.4	0.4	0.4	0.4	0.4	0.4	0.5	0.4	0.4	0.4	0.4	0.4	0.4	0.4	1.0



```
# Building output dataframe
SequenceMatchDF = pd.DataFrame(np.nan,index = ModelList , columns = ModelList)
# Iterating for each column and each row
for col in range(0,len(ModelList)):
    #print(f'Working on col number {col}')
    for row in range(0,len(ModelList)):
        #print(f' Working on row number {row}')
        similarityList = []
        # Getting a senences for each model and comparing too
        for i in range(0,len(testDF.index)):
            stringA = testDF.loc[i,ModelList[col]]
            #print(i)
            stringB = testDF.loc[i,ModelList[row]]
            similarityList.append(SequenceMatcher(None,stringA,stringB).ratio())
        # averaging the difference.
        similarityMean = statistics.mean(similarityList)
        if row == col:
            SequenceMatchDF.iloc[row,col] = statistics.mean(similarityList)
        else:
            SequenceMatchDF.iloc[row,col] = statistics.mean(similarityList)
            SequenceMatchDF.iloc[col,row] = statistics.mean(similarityList)
```

- ChatGPT had the lowest average sequence match compared to any other model
- As I increased the temperature variable, we see the average sequence match trend downward
- Models .5_2048_0_0 and .5_2048_0_1 had the lowest average match consistently.

SENTIMENT ANALYSIS



	ChatGPT discription	0_2048_0_0	0_2048_0_1	0_2048_0_2	0_2048_1_0	0_2048_1_1	0_2048_1_2	0_2048_2_0	0_2048_2_1	0_2048_2_2	0.1_2048_0_0	0.1_2048_0_1	0.1_2048_0_2	0.1_2048_1_0	0.1_2048_1_2	0.1_2048_2_0	0.1_2048_2_1	0.1_2048_2_2	0.5_2048_0_0	0.5_2048_0_1
neg	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
neu	0.9	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.6	0.7	0.7	0.8	0.7	0.8	0.8
pos	0.1	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.4	0.3	0.3	0.2	0.3	0.2	0.2
compound	0.8	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

- This used the nltk.sentiment package
  - SentimentIntensityAnalyzer()
- No Model had the average output be negative
- ChatGPT had the lowest average positive sentiment across all models
- The higher values of temperature had, on average, a higher neutral value
- Compound score is the number of all three scores
  - -1 most extreme negative
  - +1 most extreme positive
  - ChatGPT was the only model to have less then 1

KEY TAKEAWAYS

- Open AI language models can produce sophisticated and useable product descriptions
- The cost of building/training and experimenting with a GPT model is relatively low for the quality and consistency of the output
    - Less than $10 for 4.5K request
- The output from the GPT model does not have a negative or damaging sentiment.
- I recommend continuing this project into phase two.

THANK YOU.

QUESTIONS