EXPLORING MENTAL HEALTH USING LLMs:Comparison between ChatGPT and Gemini

Start coding or generate with AI.

```
#installing datasets
!pip install datasets
```

```
Collecting datasets
  Downloading datasets-2.19.0-py3-none-any.whl (542 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 542.0/542.0 kB 9.4 MB/s eta 0:00
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: pyarrow>=12.0.0 in /usr/local/lib/python3.10/di
Requirement already satisfied: pyarrow-hotfix in /usr/local/lib/python3.10/dis
Collecting dill<0.3.9,>=0.3.0 (from datasets)
  Downloading dill-0.3.8-py3-none-any.whl (116 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 116.3/116.3 kB 11.3 MB/s eta 0:0
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.10/dist-
Collecting xxhash (from datasets)
  Downloading xxhash-3.4.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.1/194.1 kB 10.7 MB/s eta 0:0
Collecting multiprocess (from datasets)
  Downloading multiprocess-0.70.16-py310-none-any.whl (134 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 13.4 MB/s eta 0:0
Requirement already satisfied: fsspec[http]<=2024.3.1,>=2023.1.0 in /usr/local
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packa
Collecting huggingface-hub>=0.21.2 (from datasets)
  Downloading huggingface_hub-0.22.2-py3-none-any.whl (388 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 388.9/388.9 kB 14.6 MB/s eta 0:0
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/d
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.1
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dis
Requirement already satisfied: async-timeout<5.0,>=4.0 in /usr/local/lib/pytho
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/py
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/pyth
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dis
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-pack
Installing collected packages: xxhash, dill, multiprocess, huggingface-hub, da
  Attempting uninstall: huggingface-hub
    Found existing installation: huggingface-hub 0.20.3
    Uninstalling huggingface-hub-0.20.3:
      Successfully uninstalled huggingface-hub-0.20.3
Successfully installed datasets-2.19.0 dill-0.3.8 huggingface-hub-0.22.2 multi
```

1. Data Collection

```
 # Loading dataset

import pandas as pd

from datasets import load_dataset

mentalhealth_dataset = load_dataset("alexandreteles/mental-health-conversational-

mentalhealth_dataset
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: Us
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access pu
  warnings.warn(
```

Downloading readme: 100% 274/274 [00:00<00:00, 17.5kB/s]

Downloading data: 100% 21.5k/21.5k [00:00<00:00, 92.7kB/s]

Generating train split: 100% 661/661 [00:00<00:00, 10897.94 examples/s]

```
DatasetDict({
    train: Dataset({
        features: ['Context', 'Knowledge', 'Response'],
        num_rows: 661
    })
})
```

```
#reading dataset using pandas

df=pd.DataFrame(mentalhealth_dataset['train'])
df
```

	Context	Knowledge	Response
0	Hi	greeting	Hello there. Tell me how are you feeling today?
1	Hi	greeting	Hi there. What brings you here today?
2	Hi	greeting	Hi there. How are you feeling today?
3	Hi	greeting	Great to see you. How do you feel currently?
4	Hi	greeting	Hello there. Glad to see you're back. What's g...
...	...	...	...
656	How do I know if I'm unwell?	fact-29	If your beliefs , thoughts , feelings or behav...
657	How can I maintain social connections? What if...	fact-30	A lot of people are alone right now, but we do...
658	What's the difference between anxiety and stress?	fact-31	Stress and anxiety are often used interchange...

```
# downloading dataset as csv

df.to_csv("mental_health_conversational_data.csv", index=False)
df
```

	Context	Knowledge	Response
0	Hi	greeting	Hello there. Tell me how are you feeling today?
1	Hi	greeting	Hi there. What brings you here today?
2	Hi	greeting	Hi there. How are you feeling today?
3	Hi	greeting	Great to see you. How do you feel currently?
4	Hi	greeting	Hello there. Glad to see you're back. What's g...
...	...	...	...
656	How do I know if I'm unwell?	fact-29	If your beliefs , thoughts , feelings or behav...
657	How can I maintain social connections? What if...	fact-30	A lot of people are alone right now, but we do...
658	What's the difference between anxiety and stress?	fact-31	Stress and anxiety are often used interchange...

```
#checking rows and columns of dataset
df.shape
```

```
(661, 3)
```

2. Data Processing

Double-click (or enter) to edit

```
# checking for null values

df.isnull().sum()
```

```
Context      0
Knowledge    0
Response     0
dtype: int64
```

```
# Removed empty strings and none values
df.replace("",None,inplace=True)
df.dropna(subset=['Context','Response'],inplace=True)
df
```

	Context	Knowledge	Response
0	Hi	greeting	Hello there. Tell me how are you feeling today?
1	Hi	greeting	Hi there. What brings you here today?
2	Hi	greeting	Hi there. How are you feeling today?
3	Hi	greeting	Great to see you. How do you feel currently?
4	Hi	greeting	Hello there. Glad to see you're back. What's g...
...	...	...	...
656	How do I know if I'm unwell?	fact-29	If your beliefs , thoughts , feelings or behav...
657	How can I maintain social connections? What if...	fact-30	A lot of people are alone right now, but we do...
658	What's the difference between anxiety and stress?	fact-31	Stress and anxiety are often used interchanges...

3. Exploratory Data Analysis

```
#checking for most frequent words in both contexts and responses
from wordcloud import WordCloud
```

```
import matplotlib.pyplot as plt

# Load your dataset
# Replace 'your_dataset.csv' with the path to your dataset file


# Join all contexts and responses into single strings
all_contexts = ' '.join(df['Context'].dropna())
all_responses = ' '.join(df['Response'].dropna())

# Create word clouds for contexts and responses
context_wordcloud = WordCloud(width=800, height=400, background_color='white').ge
response_wordcloud = WordCloud(width=800, height=400, background_color='white').g

# Plot word clouds
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.imshow(context_wordcloud, interpolation='bilinear')
plt.title('Context Word Cloud')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(response_wordcloud, interpolation='bilinear')
plt.title('Response Word Cloud')
plt.axis('off')

plt.show()
```

```
# value count  for knowledge column

knowledge_count = df['Knowledge'].value_counts()
knowledge_count
```

```
Knowledge
casual              66
greeting            60
about               48
default             40
goodbye             32
                    ..
neutral-response     1
skill                1
pandora-useful       1
morning              1
fact-11              1
Name: count, Length: 79, dtype: int64
```

```
# knowledge_count gretaer than 20

knowledge_count[knowledge_count > 20]
```

```
Knowledge
casual      66
greeting    60
about       48
default     40
goodbye     32
sad         32
done        25
help        21
happy       21
Name: count, dtype: int64
```

4. ChatGPT Response Generation

```
#installing openai
!pip install openai
```

```
Collecting openai
  Downloading openai-1.23.2-py3-none-any.whl (311 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 311.2/311.2 kB 6.0 MB/s eta 0:00
Requirement already satisfied: anyio<5,>=3.5.0 in /usr/local/lib/python3.10/di
Requirement already satisfied: distro<2,>=1.7.0 in /usr/lib/python3/dist-packa
Collecting httpx<1,>=0.23.0 (from openai)
  Downloading httpx-0.27.0-py3-none-any.whl (75 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 75.6/75.6 kB 8.5 MB/s eta 0:00:0
Requirement already satisfied: pydantic<3,>=1.9.0 in /usr/local/lib/python3.10
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: tqdm>4 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: typing-extensions<5,>=4.7 in /usr/local/lib/pyt
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dis
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packa
Collecting httpcore==1.* (from httpx<1,>=0.23.0->openai)
  Downloading httpcore-1.0.5-py3-none-any.whl (77 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 77.9/77.9 kB 9.1 MB/s eta 0:00:0
Collecting h11<0.15,>=0.13 (from httpcore==1.*->httpx<1,>=0.23.0->openai)
  Downloading h11-0.14.0-py3-none-any.whl (58 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.3/58.3 kB 7.1 MB/s eta 0:00:0
Requirement already satisfied: annotated-types>=0.4.0 in /usr/local/lib/python
Requirement already satisfied: pydantic-core==2.18.1 in /usr/local/lib/python3
Installing collected packages: h11, httpcore, httpx, openai
Successfully installed h11-0.14.0 httpcore-1.0.5 httpx-0.27.0 openai-1.23.2
```

```
#checking response for one context
import os

from openai import OpenAI

client = OpenAI(
    # This is the default and can be omitted
    api_key="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",
)

chat_completion = client.chat.completions.create(
    messages=[
        {
            "role": "user",
            "content": "What's the difference between anxiety and stress?",
        }
    ],
    model="gpt-3.5-turbo",
)


chat_completion

     ChatCompletion(id='chatcmpl-9G832Ui44TDYRQVe45kTaPiWbJL5W', choices=
     [Choice(finish_reason='stop', index=0, logprobs=None,
     message=ChatCompletionMessage(content='Anxiety is a feeling of worry, fear,
     or unease about a future event or outcome. It is typically a response to
     something perceived as threatening or potentially harmful. Stress, on the
     other hand, is a physical or emotional response to external pressures or
     demands. It can be caused by a variety of factors, such as work,
     relationships, or financial issues. While anxiety is a specific feeling of
     fear or worry, stress is a more general feeling of being overwhelmed or
     unable to cope with a situation.', role='assistant', function_call=None,
     tool_calls=None))], created=1713631212, model='gpt-3.5-turbo-0125',
     object='chat.completion', system_fingerprint='fp_c2295e73ad',
     usage=CompletionUsage(completion_tokens=101, prompt_tokens=16,
     total_tokens=117))


# using gpt-3.5-turbo generating chatgpt response for each context

# Create a new OpenAI client instance
client = OpenAI(api_key="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")

# Initialize an empty list to store the chat responses
chatgpt_responses = []
```

```
# Loop through each context in the random_responses list
for context in df['Response']:
    # Create a chat completion request with the current context as the user's mes
    chat_completion = client.chat.completions.create(
        messages=[
            {
                "role": "user",
                "content": context,
            }
        ],
        model="gpt-3.5-turbo",
    )

    # Extract the chat response from the chat completion
    chatgpt_response = chat_completion.choices[0].message.content

    # Append the chat response to the chat_responses list
    chatgpt_responses.append(chatgpt_response)

# Print the chat responses
for chatgpt_response in chatgpt_responses:
    print(chatgpt_response)
```

```
Sure thing, Pandora! What can I help you with today?
Hello Pandora, I'm feeling a bit stressed and anxious today.
Hello Pandora, it's nice to meet you. I am a human who is interested in self-i
Hello Pandora, I'm feeling a bit anxious today.
Nice to meet you, Pandora! What brings you here today?
Hello Pandora, nice to meet you! How can I assist you today?
Hello Pandora, how can I assist you today?
Hello Pandora, I'm feeling a bit overwhelmed and stressed today.
Hello Pandora, I am a human user seeking help and guidance from you. I am inte
I'm just a language model AI, so I don't have feelings or emotions. How can I
Nice to meet you, Pandora. How can I assist you today?
Hello Pandora! Nice to meet you. What can I assist you with today?
Hello Pandora! How can I assist you today?
Hello Pandora, I'm feeling a bit stressed and overwhelmed today. Thank you for
Hello, Pandora! I'm a human user seeking assistance and support. I'm looking f
Hello, Pandora. I'm feeling a bit anxious and stressed.
Hello Pandora, it's nice to meet you. How may I assist you today?
Hello, Pandora! How can I assist you today?
Sure thing, Pandora! How can I assist you today?
Hello Pandora, I'm feeling a little stressed and overwhelmed today.
Hello Pandora, nice to meet you. I am a human user seeking assistance and supp
Hello Pandora, I'm feeling a bit stressed and overwhelmed today.
Nice to meet you, Pandora! Can I help you with anything today?
Hello Pandora, nice to meet you! How can I assist you today?
```

```
Hello Pandora, nice to meet you! How can I assist you today?
Sure thing, Pandora! Is there anything I can assist you with today?
Hello Pandora, I'm feeling a bit stressed today.
I am a user interacting with you on this platform. I am curious and eager to
I'm just a language model AI and I don't have feelings. How can I assist you
Nice to meet you, Pandora! How can I assist you today?
Hi Pandora! How can I assist you today?
Sure thing, Pandora! How can I assist you today?
Hello Pandora! I'm feeling a bit stressed today.
I am a virtual assistant created to help people in need of therapy or emotiona
Hello Pandora. I'm feeling a bit overwhelmed today. I have a lot on my plate a
Nice to meet you, Pandora! How can I assist you today?
Hello Pandora! It's nice to meet you. How can I assist you today?
Hello Pandora, how can I assist you today?
Remember, it's important to prioritize your mental health and seek professiona
an artificial intelligence developer.
 to perform various tasks such as sentiment analysis, text classification, and

Through the use of neural networks and deep learning algorithms, the model was

After training, the model was able to accurately predict the sentiment of a gi

Overall, training a model on a text dataset using Deep Learning & Natural Lang
I was created by a team of developers at OpenAI.
an anonymous team of developers.
That's great to hear! What specific techniques or models did you use for your
I was created by a team of developers at OpenAI.
I am a language model AI created by OpenAI.
to perform various tasks such as text classification, text generation, sentime
I was created by a team of developers and engineers at OpenAI.
Hello! My week has been pretty good, thank you for asking. I had some producti
Hello! I'm just a computer program, so I don't have feelings like humans do. I
I'm glad you think so! I am a language model AI created to assist and engage i
I'm just a virtual assistant, so I don't have weeks or personal experiences. B
As an AI, I don't have feelings, but I'm here to assist you with whatever you
I'm glad you think so! As an AI assistant, my primary function is to assist us
As an AI, I don't experience weeks like humans do, but I'm here to assist and
```

```
print(len(chatgpt_responses))
```

```
657
```

5. Converting original Responses to Embeddings

```
# Convert 'original responses' to a list of strings
standard_response_list = df['Response'].tolist()

print(standard_response_list)
```

```
['Hello there. Tell me how are you feeling today?', 'Hi there. What brings you
```

```
# converting original responses to embeddings using bert
import transformers
import torch
# Load the BERT tokenizer and model
tokenizer = transformers.AutoTokenizer.from_pretrained("bert-base-uncased")
model = transformers.AutoModel.from_pretrained("bert-base-uncased")

# Encode the text in the responses using BERT
standard_encoded_responses = tokenizer(standard_response_list, padding=True, trun

# Get the BERT embeddings for the encoded text
with torch.no_grad():
    model_output = model(**standard_encoded_responses)
    standard_embeddings = model_output.pooler_output

# Print the shape of the embeddings
print(standard_embeddings)
```

```
tokenizer_config.json: 100%  48.0/48.0 [00:00<00:00, 3.05kB/s]
config.json: 100%  570/570 [00:00<00:00, 39.4kB/s]
vocab.txt: 100%  232k/232k [00:00<00:00, 9.60MB/s]
tokenizer.json: 100%  466k/466k [00:00<00:00, 25.1MB/s]
model.safetensors: 100%  440M/440M [00:01<00:00, 417MB/s]
tensor([[-0.9050, -0.3170, -0.7775,  ..., -0.5701, -0.6867,  0.8886],
        [-0.9451, -0.4878, -0.9316,  ..., -0.8420, -0.7578,  0.9403],
        [-0.9454, -0.4603, -0.9175,  ..., -0.7808, -0.7406,  0.9266],
        ...,
        [-0.6793, -0.5712, -0.9712,  ..., -0.9615, -0.4867,  0.3869],
        [-0.6200, -0.6038, -0.9815,  ..., -0.9313, -0.6690,  0.3167],
        [-0.6200, -0.6038, -0.9815,  ..., -0.9313, -0.6690,  0.3167]])
```

6. Converting ChatGPT Responses to Embeddings

```
# adding chatgpt_responses to dataframe

df['chatgpt_responses'] = chatgpt_responses
df
```

	Context	Knowledge	Response	chatgpt_responses
0	Hi	greeting	Hello there. Tell me how are you feeling today?	Hello! I'm just a computer program so I don't ...
1	Hi	greeting	Hi there. What brings you here today?	Hello! I'm here to assist you with any questio...
2	Hi	greeting	Hi there. How are you feeling today?	Hello! I'm just a computer program, so I don't...
3	Hi	greeting	Great to see you. How do you feel currently?	As an AI, I don't have feelings or emotions. B...
4	Hi	greeting	Hello there. Glad to see you're back. What's g...	Hello! I'm here and ready to chat. Right now, ...
...	...	...	...	...
656	How do I know if I'm unwell?	fact-29	If your beliefs , thoughts , feelings or	This could indicate that you are struggling wi

```
# Convert 'chatgpt_responses' to a list of strings

chatgpt_responses_list = df['chatgpt_responses'].tolist()

print(chatgpt_responses_list)
```

```
["Hello! I'm just a computer program so I don't have feelings, but I'm here ar
```

```
# chatgpt responses embedding using bert

# Import the necessary libraries
import transformers
import torch
# Load the BERT tokenizer and model
tokenizer = transformers.AutoTokenizer.from_pretrained("bert-base-uncased")
model = transformers.AutoModel.from_pretrained("bert-base-uncased")

# Encode the text in the responses using BERT
chatgpt_encoded_responses = tokenizer(chatgpt_responses_list, padding=True, trunc

# Get the BERT embeddings for the encoded text
with torch.no_grad():
    model_output = model(**chatgpt_encoded_responses)
    chatgpt_embeddings = model_output.pooler_output

# Print the shape of the embeddings
print(chatgpt_embeddings)
```

```
tensor([[-0.9325, -0.5729, -0.9770,  ..., -0.9012, -0.8243,  0.9313],
        [-0.9395, -0.5956, -0.9817,  ..., -0.9313, -0.8438,  0.9276],
        [-0.9352, -0.5816, -0.9873,  ..., -0.9507, -0.8229,  0.9342],
        ...,
        [-0.8952, -0.7333, -0.9807,  ..., -0.9608, -0.8024,  0.5599],
        [-0.8853, -0.7111, -0.9924,  ..., -0.9577, -0.8158,  0.5818],
        [-0.8858, -0.6367, -0.9940,  ..., -0.9273, -0.7590,  0.6535]])
```

7. Cosine Similarity score between Original responses and Chatgpt responses

```
## finding similarity score between standard responses and chatgpt_responses

from sklearn.metrics.pairwise import cosine_similarity

# Calculate the cosine similarity between the two sets of embeddings
cosine_similarity_scores_1 = cosine_similarity(standard_embeddings, chatgpt_embed

# Print the cosine similarity scores
print(cosine_similarity_scores_1)
```

```
[[0.95895934 0.95076203 0.9499899  ... 0.8611215  0.87668955 0.9094288 ]
 [0.9878885  0.98200774 0.9800472  ... 0.89887404 0.9164524  0.9467895 ]
 [0.9838684  0.97784734 0.97573936 ... 0.89491105 0.9122174  0.942554  ]
 ...
 [0.81974566 0.83403146 0.84024763 ... 0.9466718  0.9309156  0.8962947 ]
 [0.8324841  0.84710515 0.85193133 ... 0.95353323 0.9434675  0.9091039 ]
 [0.8324841  0.84710497 0.85193133 ... 0.95353323 0.9434675  0.9091039 ]]
```

```
#calculating overall (average) similarity for standard and chatgpt responses
mean_cosine_similarity_1 = cosine_similarity_scores_1.mean().item()
mean_cosine_similarity_1
```

```
0.9343438744544983
```

Double-click (or enter) to edit

The Cosine similarity score between original responses and ChatGPT response is 93 % which is good.

8. Gemini Response Generation

```
#installing generative AI
!pip install -q -U google-generativeai
```

```
                                        146.8/146.8 kB 3.3 MB/s eta 0:0(
                                        664.5/664.5 kB 23.2 MB/s eta 0:(
```

```
#configuring gemini api key
import google. generativeai as genai
API_Key= 'AIzaSyAb427u9nVJy4qNH95nnro-ZpbvAwrwppM'
genai.configure(api_key=API_Key)


 #safety settings for excluding harm content
safety_settings = [
    {
        "category": "HARM_CATEGORY_DANGEROUS",
        "threshold": "BLOCK_NONE",
    },
    {
        "category": "HARM_CATEGORY_HARASSMENT",
        "threshold": "BLOCK_NONE",
    },
    {
        "category": "HARM_CATEGORY_HATE_SPEECH",
        "threshold": "BLOCK_NONE",
    },
    {
        "category": "HARM_CATEGORY_SEXUALLY_EXPLICIT",
        "threshold": "BLOCK_NONE",
    },
    {
        "category": "HARM_CATEGORY_DANGEROUS_CONTENT",
        "threshold": "BLOCK_NONE",
    },
]
#checking response for single context
context = "Hi"
model = genai.GenerativeModel('gemini-pro')
generated_response = model.generate_content(context, safety_settings=safety_setti
print(generated_response)
```

```
Hello there. How can I assist you today?
```

```
# generating gemini responses using gemini-pro
generated_responses = []
for context in df['Context']:
    # Check if the context is not empty
    if context.strip():
        response = model.generate_content(context, safety_settings=safety_setting
        generated_responses.append(response)
    else:
        continue
        # generated_responses.append("Empty context")
df['generated_response'] = generated_responses
```

```
df['generated_response']
```

```
0      response:\nGenerateContentResponse(\n    done=...
1      response:\nGenerateContentResponse(\n    done=...
2      response:\nGenerateContentResponse(\n    done=...
3      response:\nGenerateContentResponse(\n    done=...
4      response:\nGenerateContentResponse(\n    done=...
                              ...
656    response:\nGenerateContentResponse(\n    done=...
657    response:\nGenerateContentResponse(\n    done=...
658    response:\nGenerateContentResponse(\n    done=...
659    response:\nGenerateContentResponse(\n    done=...
660    response:\nGenerateContentResponse(\n    done=...
Name: generated_response, Length: 657, dtype: object
```

```
#looping all the responses from gemini
for each in generated_responses:
  print(each)
```

```
response:
GenerateContentResponse(
    done=True,
    iterator=None,
    result=glm.GenerateContentResponse({'candidates': [{'content': {'parts':
)
response:
GenerateContentResponse(
    done=True,
    iterator=None,
    result=glm.GenerateContentResponse({'candidates': [{'content': {'parts':
)
response:
```

```
GenerateContentResponse(
    done=True,
    iterator=None,
    result=glm.GenerateContentResponse({'candidates': [{'content': {'parts':
)
response:
GenerateContentResponse(
    done=True,
    iterator=None,
    result=glm.GenerateContentResponse({'candidates': [{'content': {'parts':
)
response:
GenerateContentResponse(
    done=True,
    iterator=None,
    result=glm.GenerateContentResponse({'candidates': [{'content': {'parts':
)
response:
GenerateContentResponse(
    done=True,
    iterator=None,
    result=glm.GenerateContentResponse({'candidates': [{'content': {'parts':
)
response:
GenerateContentResponse(
    done=True,
    iterator=None,
    result=glm.GenerateContentResponse({'candidates': [{'content': {'parts':
)
response:
GenerateContentResponse(
    done=True,
    iterator=None,
    result=glm.GenerateContentResponse({'candidates': [{'content': {'parts':
)
response:
GenerateContentResponse(
    done=True,
    iterator=None,
    result=glm.GenerateContentResponse({'candidates': [{'content': {'parts':
)
response:
GenerateContentResponse(
    done=True,
    iterator=None,
    result=glm.GenerateContentResponse({'candidates': [{'content': {'parts':
)
```

```
#gemini responses have candidate keys so handling them
gemini_responses = []
for candidate in generated_responses:
    try:
        gemini_responses.append(candidate.text)
    except:
        # print("booo yeah",candidate)
        gemini_responses.append("No response found")



gemini_responses
```

```
e of information. Check multiple sources to get a well-rounded view
opic. This will help you avoid bias and ensure that you\'re getting
information.\n4. **Be critical.** Don\'t just accept everything you
ace value. Be critical of the information you find and ask yourself
 like: Who wrote this? What is their bias? Is this information up-
n5. **Cite your sources.** When you use information from other
be sure to cite your sources. This will help you avoid plagiarism
credit to the original authors.',
mations for Self-Love and Acceptance**\n\n* I am worthy of love and
rom myself and others.\n* I accept and embrace my strengths and
s.\n* I am beautiful and unique in my own way.\n* I am not defined
takes or failures.\n* I am capable and deserving of success.\n* I am
d resilient.\n* I am worthy of happiness and fulfillment.\n* I
yself for my past mistakes.\n* I am worthy of loving and being
 I am enough just as I am.\n* I am grateful for my life and all that
* I choose to focus on the positive aspects of my life.\n* I am
more confident and self-assured every day.\n* I am creating a life
uly love.\n\n**Affirmations for Positive Thinking**\n\n* I am
f achieving great things.\n* I am surrounded by positive people who
e.\n* I am grateful for the opportunities that come my way.\n* I am
a positive and fulfilling life for myself.\n* I am making progress
y goals every day.\n* I am optimistic and believe in a bright
* I am focused on the solutions rather than the problems.\n* I am
from my mistakes and growing as a person.\n* I am surrounded by
d abundance.\n* I am creating a life that I am excited to live.\n* I
ng to be happy and positive every day.\n* I am making a difference
rld, one step at a time.\n* I am grateful for the many blessings in
n\n**Affirmations for Success**\n\n* I am confident and capable of
 my goals.\n* I am motivated and determined to succeed.\n* I am
and ready for any challenge that comes my way.\n* I am surrounded by
e people who believe in me.\n* I am taking inspired action every
 am creating a successful and prosperous life for myself.\n* I am
 my goals with ease and grace.\n* I am grateful for the abundance
ss in my life.\n* I am making a positive impact on the world.\n* I
 a life of purpose and fulfillment.\n* I am achieving my dreams and
ns.\n* I am proud of my accomplishments.\n* I am constantly learning
ng.\n\n**Affirmations for Abundance**\n\n* I am open to receiving
```

```
ng:\n\n**Affirmations for Abundance:**\n\n* I am open to receiving
 in all areas of my life.\n* I am worthy of prosperity and
* I am creating a life of abundance for myself and others.\n* The
is conspiring to bring me all that I desire.\n* I am grateful for
ance that I already have.\n* I am attracting more and more abundance
ife.\n* I am living in a world of plenty.\n* I am surrounded by
 and prosperity.\n* I am choosing to focus on the abundance in my
I am using my abundance to help others.\n* I am creating a positive
ous world.\n* I am open to receiving all the good that life has to
 I am worthy of an abundant and fulfilling life.',
its of Tai Chi for Seniors:**\n\n* **Improved balance:** Tai chi
slow, gentle movements that help to enhance coordination, balance,
al awareness, reducing the risk of falls.\n* **Increased strength
bility:** The movements in tai chi require the use of various muscle
mproving both strength and flexibility.\n* **Pain reduction:** Tai
een shown to be effective in reducing pain and stiffness associated
oarthritis and other chronic conditions.\n* **Improved
cular health:** While it is not an intense workout, tai chi can
irculation and reduce blood pressure.\n* **Mental well-being:** Tai
tes relaxation, reduces stress, and improves cognitive function. It
be a beneficial form of social interaction.\n* **Low-impact
** Tai chi is a low-impact activity that can be done by seniors
s of their fitness level or physical limitations.\n\n**Getting
ith Tai Chi for Seniors:**\n\n* **Consult a healthcare
nal:** Before starting any exercise program, it is advisable to
```

```
#checking datatype of gemini responses

print(type(gemini_responses))
```

```
<class 'list'>
```

```
#checking length of gemini response

print(len(gemini_responses))
```

```
657
```

```
# Adding gemini responses to dataset

df['gemini_responses'] = gemini_responses
df.head()
```

	Context	Knowledge	Response	chatgpt_responses	generated_resp
0	Hi	greeting	Hello there. Tell me how are you feeling today?	Hello! I'm just a computer program so I don't ...	response:\nGenerateContentRespo do
1	Hi	greeting	Hi there. What brings you here today?	Hello! I'm here to assist you with any questio...	response:\nGenerateContentRespo do
			Hi there. How are	Hello! I'm just a	

Next steps: Generate code with df View recommended plots

```
#converting gemini responses to list
gemini_responses_list = df['gemini_responses'].tolist()
gemini_responses_list
```

```
challenging situations.\n* With repeated exposure, your fear will gradually
decrease.\n\n**Relaxation Techniques:**\n\n* Practice relaxation methods such
as deep breathing exercises, meditation, or yoga.\n* These techniques can
help calm your nervous system and reduce anxiety.\n\n**Mindfulness:**\n\n*
Pay attention to the present moment without judgment.\n* Focus on what you
can control and let go of worries about the future.\n\n**Social
Support:**\n\n* Talk to a trusted friend, family member, therapist, or
support group about your fears.\n* Sharing your feelings can help you feel
less alone and gain support.\n\n**Professional Help:**\n\n* If your fears are
persistent and significantly impacting your life, consider seeking
professional help.\n* A therapist can provide personalized guidance, support,
and coping mechanisms.\n\n**Self-Care:**\n\n* Prioritize your physical and
mental well-being.\n* Get enough sleep, eat healthy, and engage in activities
that bring you joy.\n* Taking care of yourself can help you better manage
your fears.\n\n**Remember that:**\n\n* You are not alone in experiencing
fear.\n* Fears can be managed and overcome with effort and support.\n* You
have the strength and resilience to face your fears and create a more
fulfilling life for yourself.',
```

'I am so sorry to hear that your mother has passed away. Losing a loved one is always difficult, and losing a parent can be especially painful. I offer my deepest condolences during this difficult time.\n\nHere are some resources that may be helpful to you:\n\n* **National Suicide Prevention Lifeline:** 1-800-273-TALK (8255)\n* **Crisis Text Line:** Text HOME to 741741\n* **American Foundation for Suicide Prevention:** https://afsp.org\n* **National Alliance on Mental Illness (NAMI):** https://www.nami.org\n\nPlease know that you are not alone. There are people who care about you and want to help you through this difficult time.',
 "I am so sorry to hear about the loss of your mother. Losing a loved one is one of the most difficult experiences one can go through. It's important to allow yourself time to grieve and process your emotions. Know that you are not alone and there are people who care about you and want to support you during this difficult time. If you need someone to talk to or need any assistance, please don't hesitate to reach out.",
 'I am very sorry to hear that your mother has passed away. My deepest condolences to you and your family during this difficult time. Losing a loved one is never easy, and it can be especially challenging when it is someone as close as a mother. Please know that you are not alone in your grief, and there are people who care about you and want to support you. If you need someone to talk to or need assistance, please reach out to a trusted friend, family member, or mental health professional.',
 "I am so sorry to hear about the loss of your brother. My deepest condolences go out to you and your family during this difficult time. The loss of a loved one is always painful, but losing a sibling can be especially devastating. Please know that there are people who care about you and want to help. If you need someone to talk to or just need a shoulder to cry on, please don't hesitate to reach out.",
 "I am sorry to hear that. Losing a loved one is never easy, but losing a sibling can be especially difficult. I can't imagine what you must be going through right now. If you need someone to talk to, please don't hesitate to reach out to me.",
 "I am so sorry to hear about the loss of your brother. My deepest condolences to you and your family during this difficult time. Please know that you are not alone in your grief, and there are people who care about you and want to help. If you need someone to talk to or need assistance, please don't hesitate to reach out for support.",
 'I am sorry to hear that. I know that losing a loved one is never easy. It is important to remember that you are not alone and that there are people who care about you. If you need someone to talk to or need support, please reach out to a friend, family member, or therapist.',
 'I\'m so sorry for your loss. Losing a loved one is never easy, and it can

9. Converting Gemini responses to embeddings

```
#embedding gemini responses
import transformers
import torch
# Load the BERT tokenizer and model
tokenizer = transformers.AutoTokenizer.from_pretrained("bert-base-uncased")
model = transformers.AutoModel.from_pretrained("bert-base-uncased")

# Encode the text in the responses using BERT
gemini_encoded_responses = tokenizer(gemini_responses_list, padding=True, truncat

# Get the BERT embeddings for the encoded text
with torch.no_grad():
    model_output = model(**gemini_encoded_responses)
    gemini_embeddings = model_output.pooler_output


# gemini embeddings
print(gemini_embeddings)
```

```
tensor([[-0.8966, -0.3980, -0.8284,  ..., -0.7591, -0.7610,  0.9050],
        [-0.9006, -0.3840, -0.8008,  ..., -0.7328, -0.7512,  0.9121],
        [-0.8966, -0.3980, -0.8284,  ..., -0.7591, -0.7610,  0.9050],
        ...,
        [-0.5288, -0.6660, -0.9913,  ..., -0.9870, -0.5686,  0.0077],
        [-0.5616, -0.6662, -0.9896,  ..., -0.9633, -0.7388,  0.1805],
        [-0.8736, -0.7195, -0.9603,  ..., -0.8491, -0.8212,  0.6462]])
```

```
# shape(rows and columns) of the embeddings
print(gemini_embeddings.shape)
```

```
torch.Size([657, 768])
```

10. Cosine Similarity score between original and gemini responses

```
# finding similarity score between responses and gemini_responses

from sklearn.metrics.pairwise import cosine_similarity

# Calculate the cosine similarity between the two sets of embeddings
cosine_similarity_scores_2 = cosine_similarity(standard_embeddings, gemini_embedd

# Print the cosine similarity scores
print(cosine_similarity_scores_2)
```

```
[[0.9905597  0.99202466 0.9905597  ... 0.67092466 0.7519696  0.88011366]
 [0.98959064 0.98811376 0.98959064 ... 0.7305476  0.8020142  0.9211122 ]
 [0.9895622  0.9885932  0.9895622  ... 0.7224069  0.7949421  0.9168322 ]
 ...
 [0.77813643 0.7687261  0.77813643 ... 0.960749   0.96421707 0.8937079 ]
 [0.7862462  0.77657616 0.7862462  ... 0.96443605 0.97721994 0.9090022 ]
 [0.7862464  0.77657634 0.7862464  ... 0.96443635 0.97721994 0.9090022 ]]
```

```
#overall(average) similarity of gemini embeddings
mean_cosine_similarity_2 = cosine_similarity_scores_2.mean().item()
mean_cosine_similarity_2
```

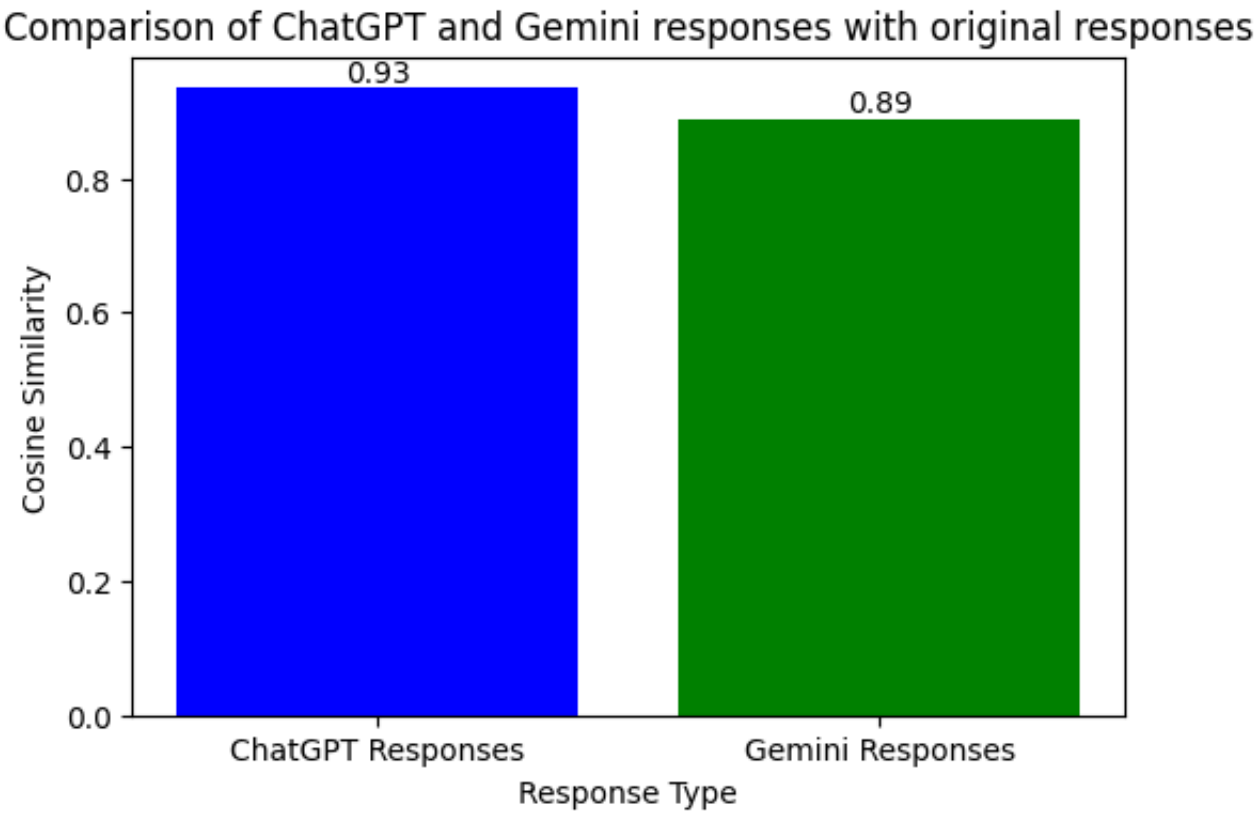
```
0.8885164856910706
```

The overall Cosine score between original responses and Gemini responses is 88.61% which is also good but compared to chatgpt performance the gemini performance is not great.

11. Data Visualization of comparing performance of ChatGPT and Gemini

```
# plotting graph for comparing  of chatgpt and gemini using cosine similarities

import matplotlib.pyplot as plt

# Prepare data
labels = ['ChatGPT Responses', 'Gemini Responses']
cosine_similarities = [mean_cosine_similarity_1, mean_cosine_similarity_2]
```

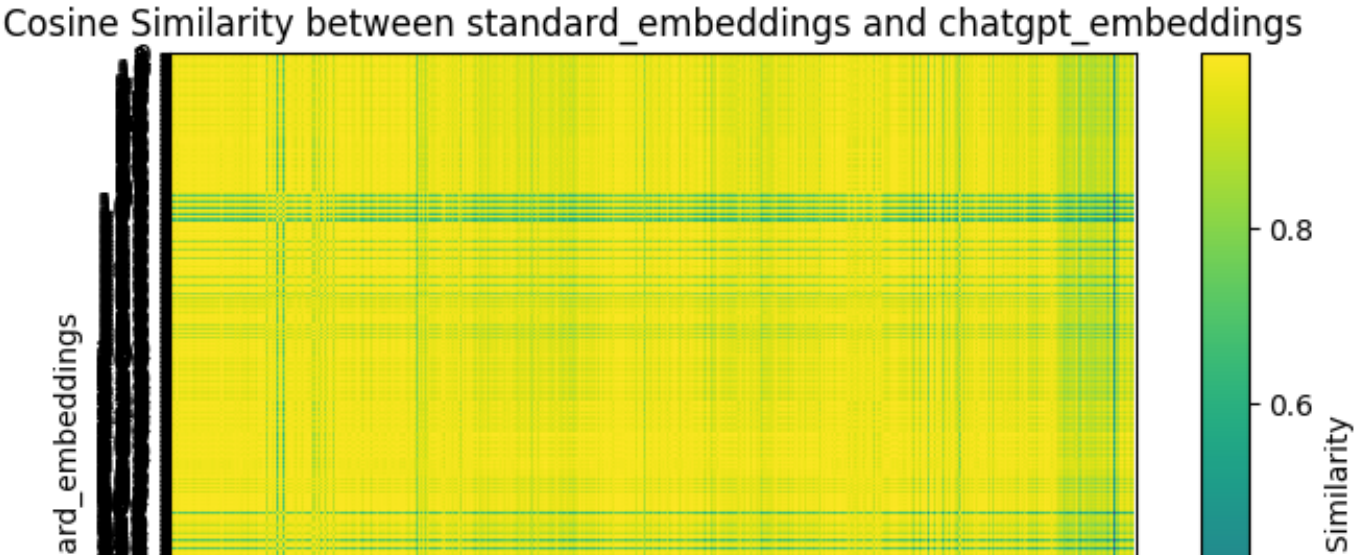
```
# Create bar chart
plt.figure(figsize=(6, 4))
bars = plt.bar(labels, cosine_similarities, color=['blue', 'green'])

# Add labels and title
plt.xlabel('Response Type')
plt.ylabel('Cosine Similarity')
plt.title('Comparison of ChatGPT and Gemini responses with original responses')

# Add percentage labels to the bars
for bar, similarity in zip(bars, cosine_similarities):
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(), f'{similarity:.
             ha='center', va='bottom')

# Show plot
plt.show()
```

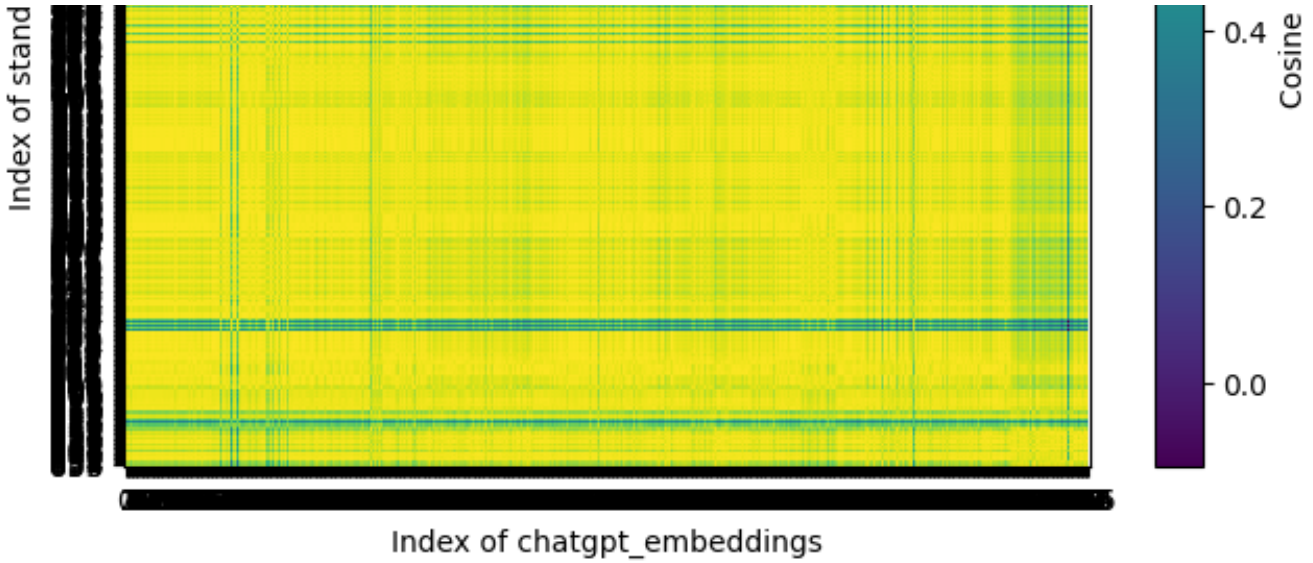


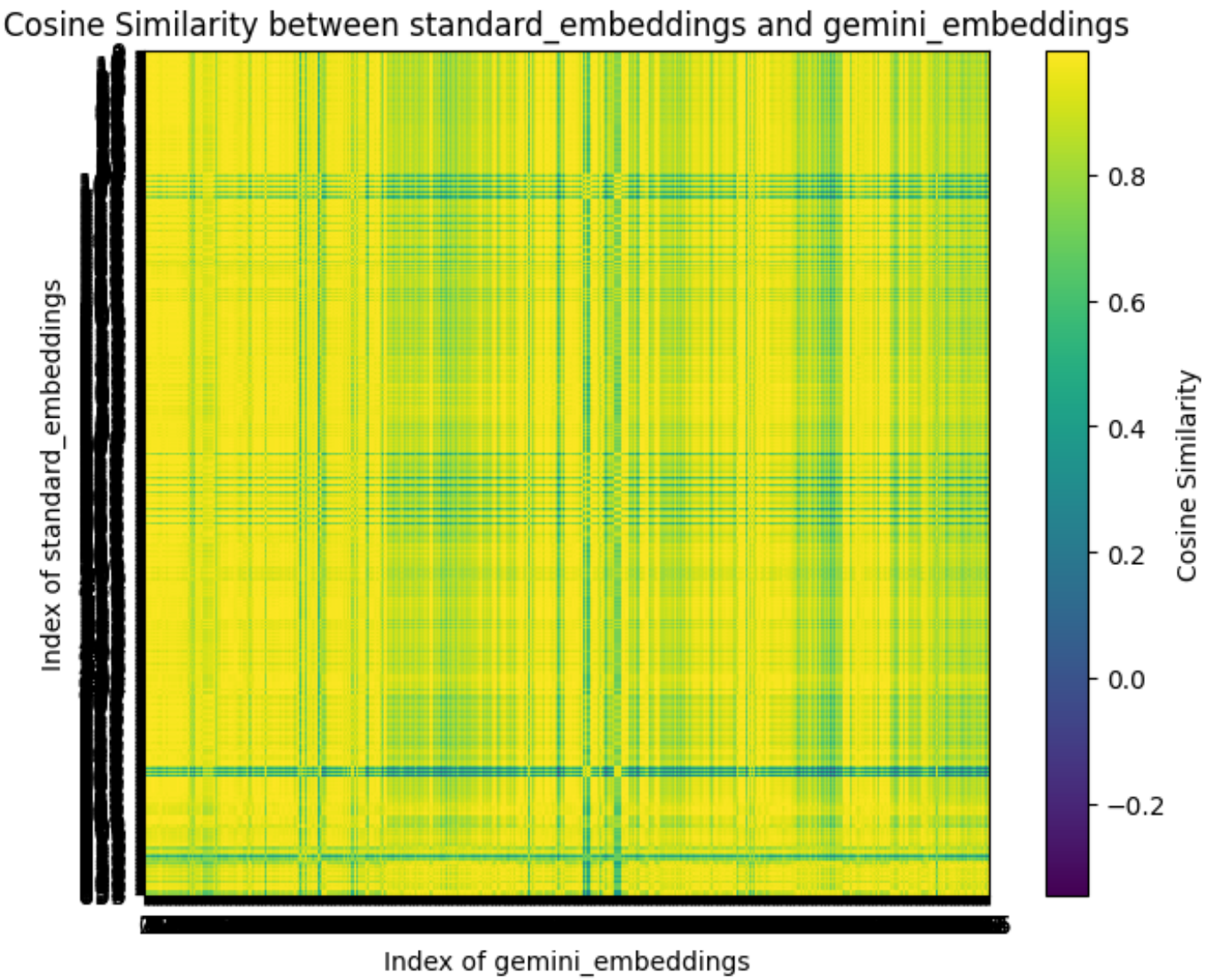
Above plot shows that ChatGPT is performing better than Gemini.

```
#plotting charts to comare embeddings of orginal, gpt and gemini
```

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity


# Calculate  cosine_similarity(standard_embeddings, chatgpt_embeddings)
cosine_sim_1 = cosine_similarity(standard_embeddings, chatgpt_embeddings)
# Plotting the cosine similarity
plt.figure(figsize=(8, 6))
plt.imshow(cosine_sim_1, cmap='viridis', interpolation='nearest')
plt.colorbar(label='Cosine Similarity')
plt.xlabel('Index of chatgpt_embeddings')
plt.ylabel('Index of standard_embeddings')
plt.title('Cosine Similarity between standard_embeddings and chatgpt_embeddings')
plt.yticks(np.arange(len(standard_embeddings)), np.arange(len(standard_embeddings
plt.xticks(np.arange(len(chatgpt_embeddings)), np.arange(len(chatgpt_embeddings))
plt.show()
# Calculate cosine similarity between each pair of points
cosine_sim_2 = cosine_similarity(standard_embeddings, gemini_embeddings)

# Plotting the cosine similarity
plt.figure(figsize=(8, 6))
plt.imshow(cosine_sim_2, cmap='viridis', interpolation='nearest')
plt.colorbar(label='Cosine Similarity')
plt.xlabel('Index of gemini_embeddings')
plt.ylabel('Index of standard_embeddings')
plt.title('Cosine Similarity between standard_embeddings and gemini_embeddings')
plt.yticks(np.arange(len(standard_embeddings)), np.arange(len(standard_embeddings
plt.xticks(np.arange(len(gemini_embeddings)), np.arange(len(gemini_embeddings)))
plt.show()
```




Index of stand
Index of chatgpt_embeddings
0.4
0.2
0.0
Cosine


Cosine Similarity between standard_embeddings and gemini_embeddings
Index of standard_embeddings
Index of gemini_embeddings
0.8
0.6
0.4
0.2
0.0
−0.2
Cosine Similarity

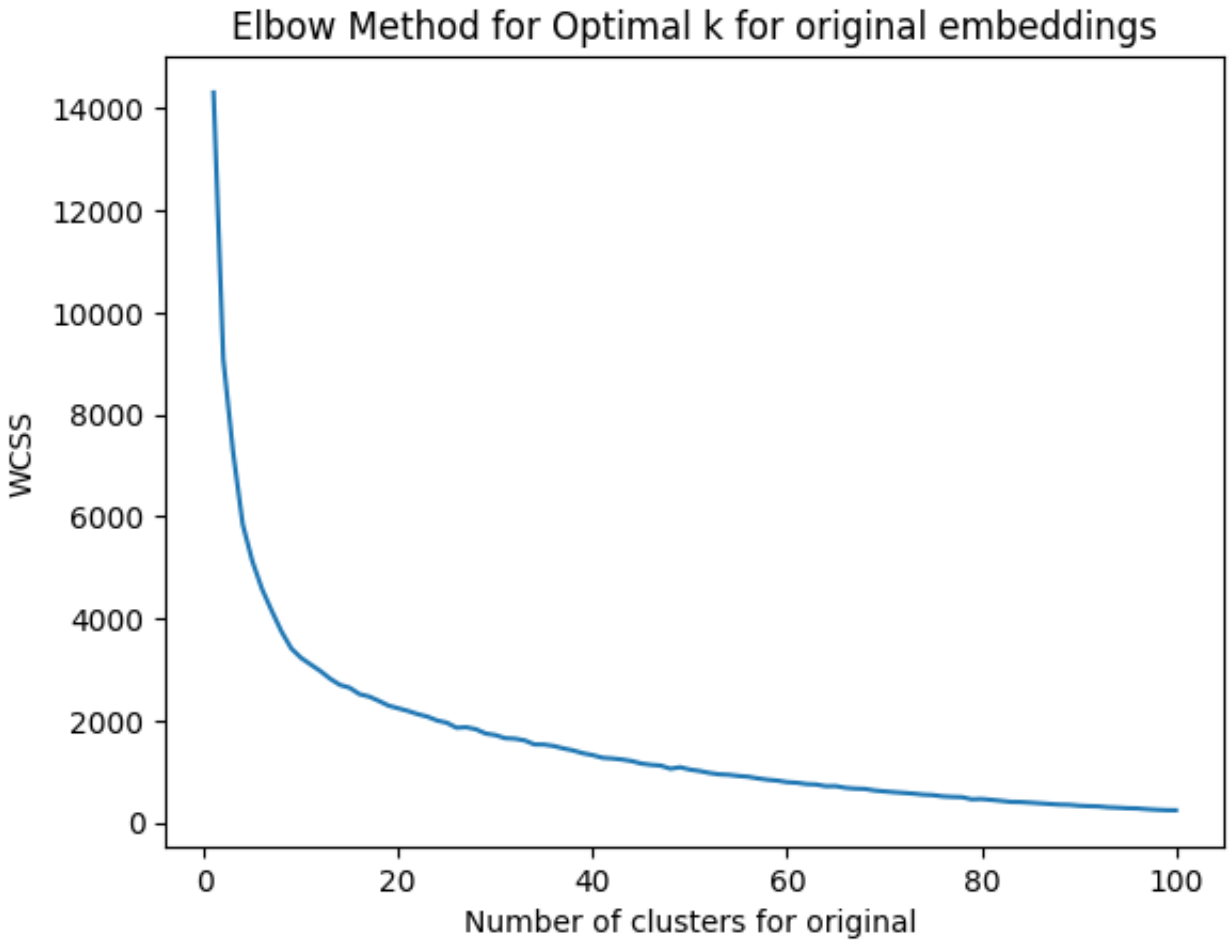
The above first chart is plotted between Embeddings of original responses and ChatGPT responses where yellow color represents that similarity score is greater than 80

The above second chart is plotted between Embeddings of original responses and Gemini responses where most of the color is yellow which represents that similarity score is greater than 80

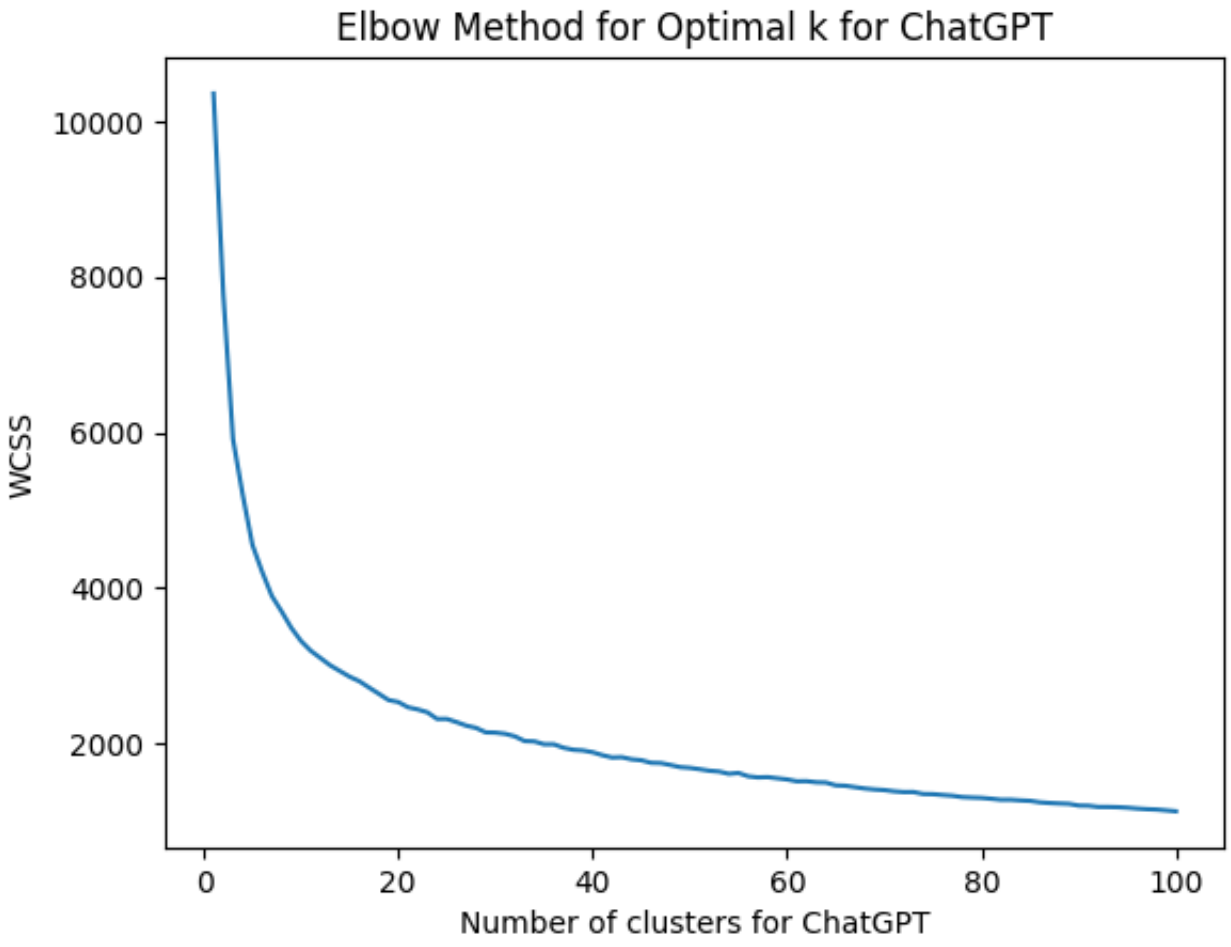
12. Finding optimal number of clusters for Original, ChatGPT and Gemini Embeddings and plotting scatter plot

```
import warnings
warnings.filterwarnings('ignore')
```

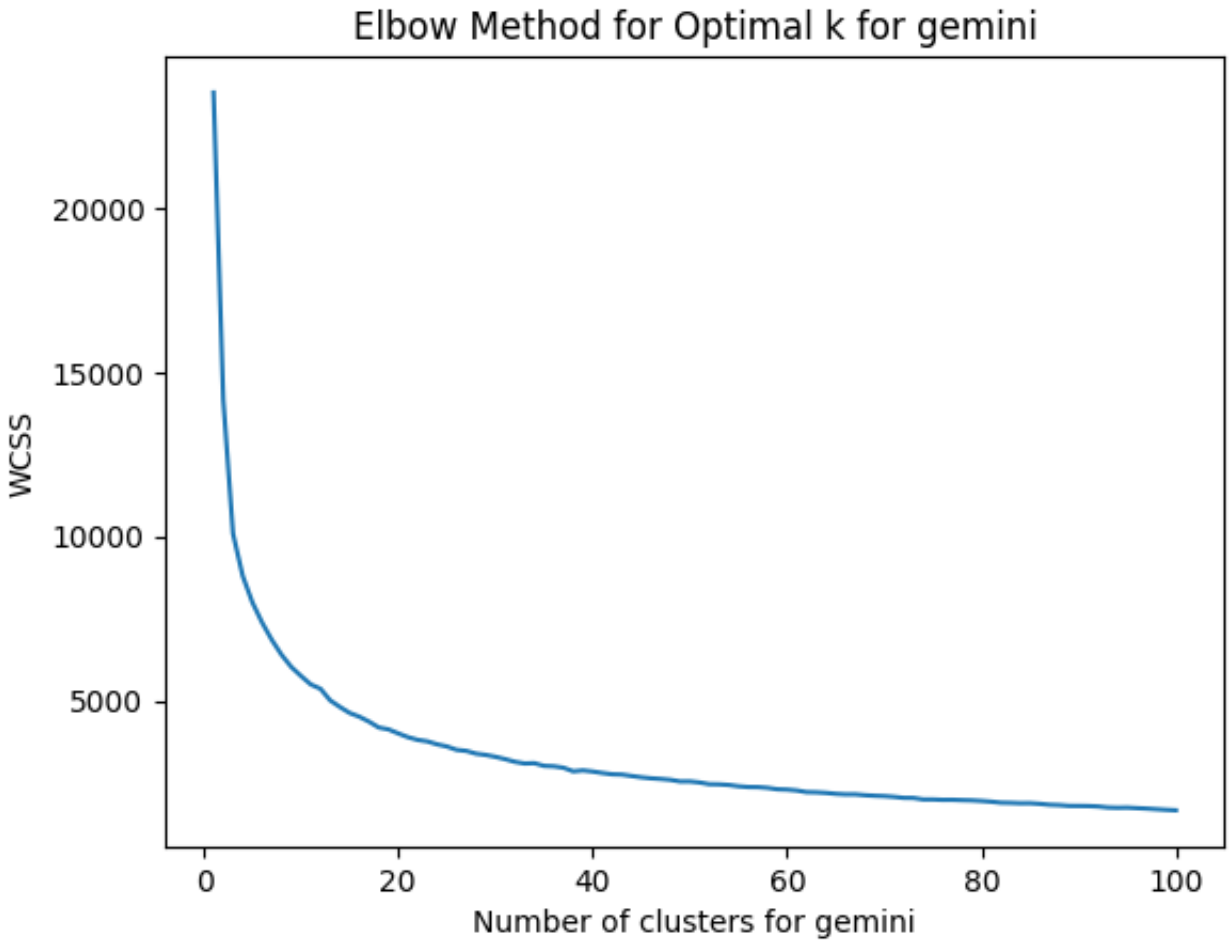
```
#finding optimal number of clusters for original response embeddings using  elbow
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans


X = standard_embeddings

wcss = []
for i in range(1, 101):  # Range from 1 to 50 clusters
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

# Plot the WCSS values
plt.plot(range(1, 101), wcss)
plt.xlabel('Number of clusters for original ')
plt.ylabel('WCSS')
plt.title('Elbow Method for Optimal k for original embeddings')
plt.show()
```


Elbow Method for Optimal k for original embeddings
WCSS
14000
12000
10000
8000
6000
4000
2000
0
0
20
40
60
80
100
Number of clusters for original

```
#finding optimal number of clusters for chatgpt response embeddings using  elbow


wcss = []
for i in range(1, 101):  # Range from 1 to 50 clusters
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(chatgpt_embeddings)
    wcss.append(kmeans.inertia_)

# Plot the WCSS values
plt.plot(range(1, 101), wcss)
plt.xlabel('Number of clusters for ChatGPT')
plt.ylabel('WCSS')
plt.title('Elbow Method for Optimal k for ChatGPT')
plt.show()
```



```
#finding optimal number of clusters for gemini response embeddings using  elbow c

wcss = []
for i in range(1, 101):  # Range from 1 to 50 clusters
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(gemini_embeddings)
    wcss.append(kmeans.inertia_)

# Plot the WCSS values
plt.plot(range(1, 101), wcss)
plt.xlabel('Number of clusters for gemini')
plt.ylabel('WCSS')
plt.title('Elbow Method for Optimal k for gemini')
plt.show()
```

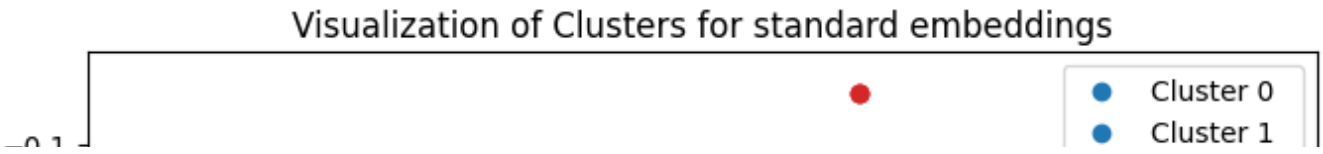


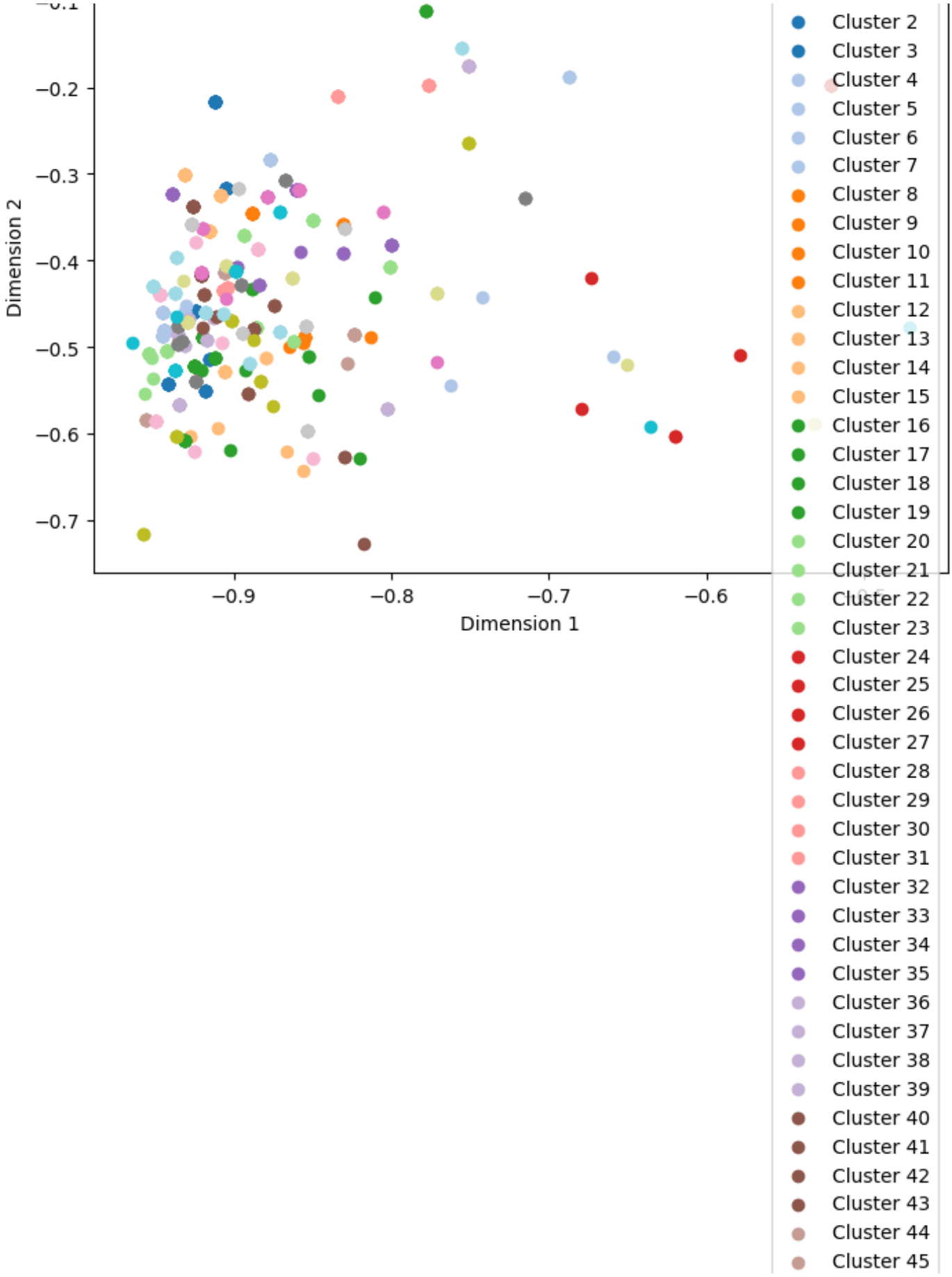
```
print(len(standard_embeddings))
```

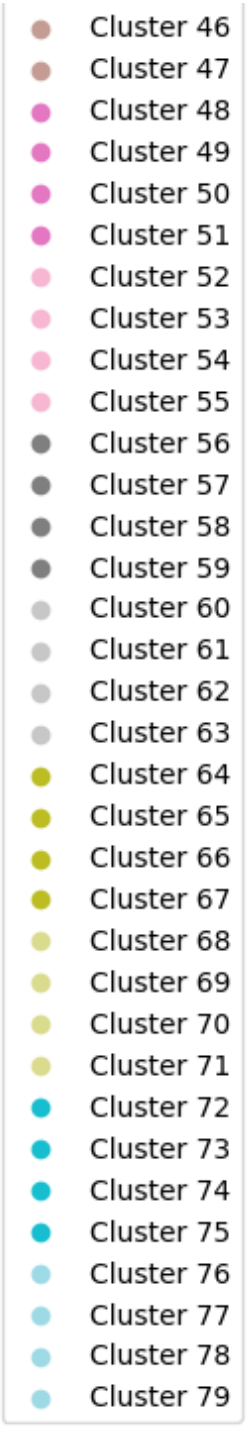
```
657
```

From the above elbow charts we can see that the line is getting straight at point 80 so we can take 80 as the optimal number of clusters

```
#clustering  and scatterplot for standard embeddings

from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import numpy as np

# Initialize K-means with desired number of clusters
kmeans = KMeans(n_clusters=80, random_state=42)

# Fit K-means to your embedding matrix
standard_cluster_labels = kmeans.fit_predict(standard_embeddings)

# Generate 80 distinct colors for clusters
colors = plt.cm.tab20(np.linspace(0, 1, 80))

# Create a dictionary mapping cluster labels to colors
color_dict = {cluster_label: color for cluster_label, color in zip(range(80), col

# Plot the clusters using the first two dimensions of the embedding matrix
plt.figure(figsize=(8, 6))
for cluster_label in range(80):
    cluster_mask = standard_cluster_labels == cluster_label
    plt.scatter(standard_embeddings[cluster_mask, 0], standard_embeddings[cluster_
                c=[color_dict[cluster_label]], label=f'Cluster {cluster_label}')

plt.title('Visualization of Clusters for standard embeddings')
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.legend()
plt.show()
```




−0.2
−0.3
−0.4
−0.5
−0.6
−0.7
Dimension 2
−0.9
−0.8
−0.7
−0.6
Dimension 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Cluster 8
Cluster 9
Cluster 10
Cluster 11
Cluster 12
Cluster 13
Cluster 14
Cluster 15
Cluster 16
Cluster 17
Cluster 18
Cluster 19
Cluster 20
Cluster 21
Cluster 22
Cluster 23
Cluster 24
Cluster 25
Cluster 26
Cluster 27
Cluster 28
Cluster 29
Cluster 30
Cluster 31
Cluster 32
Cluster 33
Cluster 34
Cluster 35
Cluster 36
Cluster 37
Cluster 38
Cluster 39
Cluster 40
Cluster 41
Cluster 42
Cluster 43
Cluster 44
Cluster 45

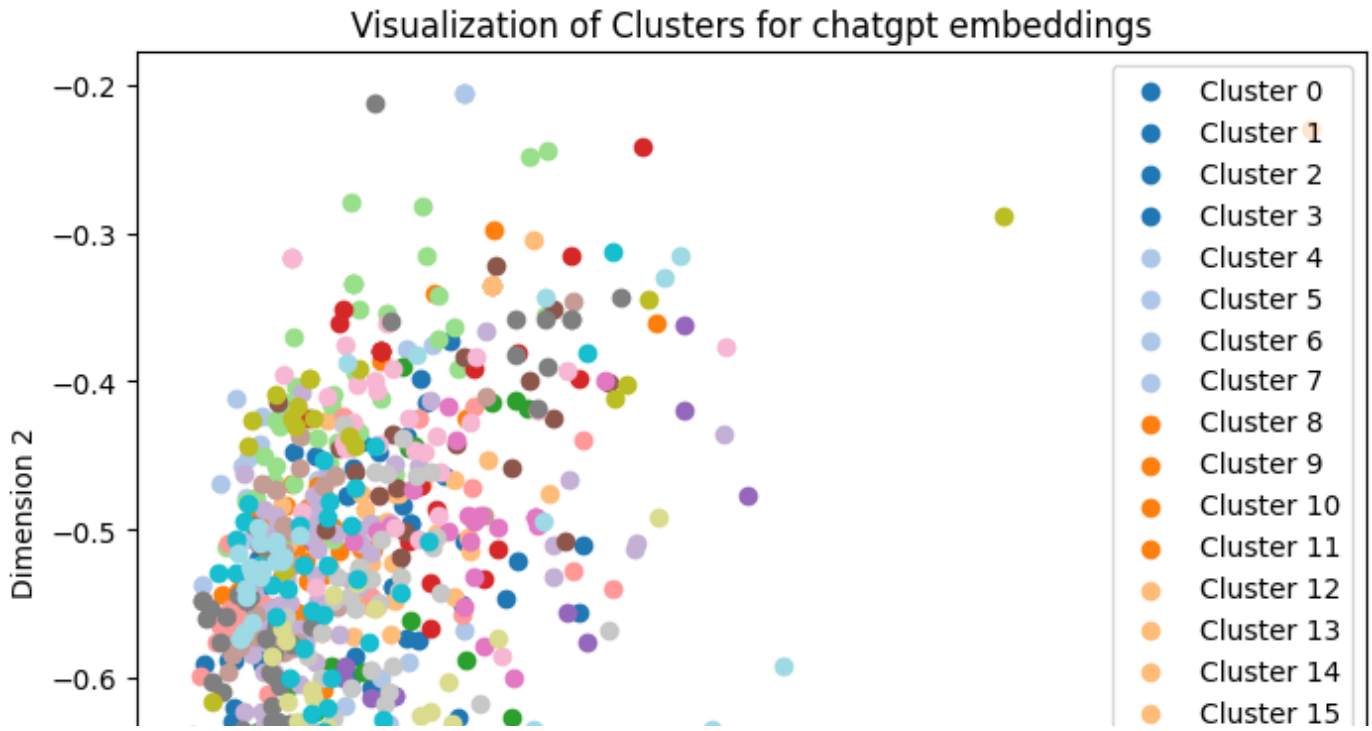

Cluster 46
Cluster 47
Cluster 48
Cluster 49
Cluster 50
Cluster 51
Cluster 52
Cluster 53
Cluster 54
Cluster 55
Cluster 56
Cluster 57
Cluster 58
Cluster 59
Cluster 60
Cluster 61
Cluster 62
Cluster 63
Cluster 64
Cluster 65
Cluster 66
Cluster 67
Cluster 68
Cluster 69
Cluster 70
Cluster 71
Cluster 72
Cluster 73
Cluster 74
Cluster 75
Cluster 76
Cluster 77
Cluster 78
Cluster 79

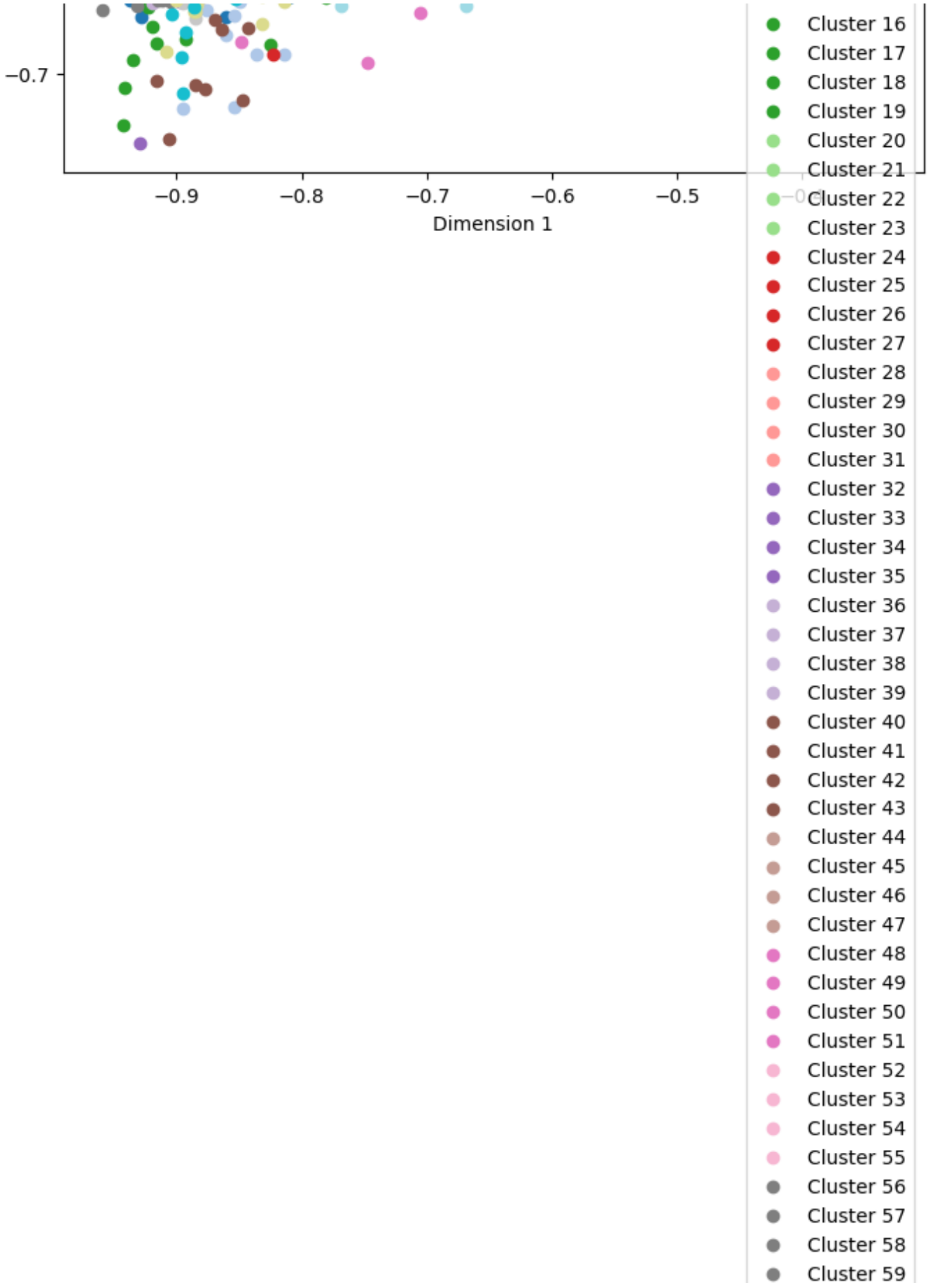
```
len(set((standard_cluster_labels)))
standard_cluster_labels
```

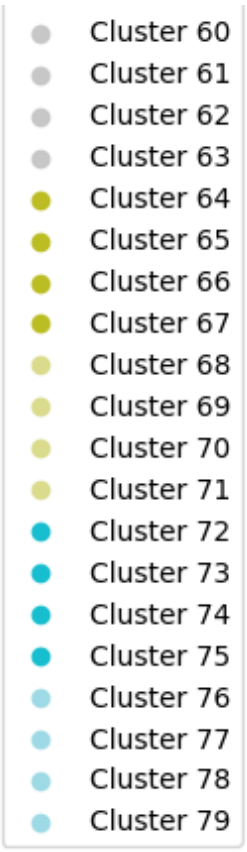
```
array([ 1,  7,  7, 21, 53,  1,  7,  7, 21, 53,  1,  7,  7, 21, 53,  1,  7,
        7, 21, 53,  1,  7,  7, 21, 53,  1,  7,  7, 21, 53,  1,  7,  7, 21,
       53,  1,  7,  7, 21, 53,  1,  7,  7, 21, 53,  1,  7,  7, 21, 53,  1,
        7,  7, 21, 53,  1,  7,  7, 21, 53, 21, 21, 21, 74, 74, 74,  0, 40,
       76, 74,  0, 40, 76, 74,  0, 40, 76, 74,  0, 40, 76, 74,  0, 40, 76,
       74,  0, 40, 76, 74,  0, 40, 76, 74,  0, 40, 76, 59, 29, 30, 57, 59,
       29, 30, 57, 59, 29, 30, 57, 59, 29, 30, 57,  7, 38,
       22, 69, 54, 52, 35, 38, 22, 69, 54, 52, 35, 38, 22, 69, 54, 52, 35,
       38, 22, 69, 54, 52, 35, 38, 22, 69, 54, 52, 35, 38, 22, 69, 54, 52,
       35, 38, 22, 69, 54, 52, 35, 38, 22, 69, 54, 52, 35, 67, 34, 63, 72,
       34, 63, 72, 34, 63, 72, 69, 39, 10, 69, 39, 10, 69, 39, 10, 48, 49,
       21, 48, 49, 21, 48, 49, 21, 48, 49, 21, 48, 49, 21, 48, 49, 21, 48,
       49, 21,  2, 79, 13, 13,  2, 79, 13, 13,  2, 79, 13, 13,  2, 79, 13,
       13,  2, 79, 13, 13,  2, 79, 13, 13,  2, 79, 13, 13,  2, 79, 13, 13,
       23, 47,  0, 31, 23, 47,  0, 31, 23, 47,  0, 31, 23, 47,  0, 31, 23,
       47,  0, 31, 79, 19, 11, 11, 79, 19, 11, 11, 79, 19, 11, 11, 79, 19,
       11, 11, 79, 19, 11, 11, 64, 56, 78, 64, 56, 78, 64, 56, 78, 64, 56,
       78, 19,  6, 46, 19,  6, 46, 19,  6, 46, 19,  6, 46, 19,  6, 46, 19,
        6, 46, 19,  6, 46, 19,  7, 28,  3, 14,  9, 19,  7, 28,  3, 14,  9,
       19,  7, 28,  3, 14,  9, 19,  7, 28,  3, 14,  9, 19,  7, 28,  3, 14,
        9, 19,  7, 28,  3, 14,  9, 19,  7, 28,  3, 14,  9, 19,  7, 28,  3,
       14,  9, 19,  7, 28,  3, 14,  9, 19,  7, 28,  3, 14,  9, 19,  7, 28,
        3, 14,  9, 53, 14, 32, 42, 53, 14, 32, 42, 78, 32,  2, 42, 78, 32,
        2, 42, 78, 32,  2, 42, 78, 32,  2, 42, 23, 36, 23, 36, 23, 36, 23,
       36, 23, 36, 23, 36, 56, 51, 56, 56, 51, 56, 56, 51, 56, 56, 51, 56,
       16, 60, 16, 16, 60, 16, 16, 60, 16, 16, 60, 16, 16, 60, 16, 16, 60,
       16, 75, 44, 56, 75, 44, 56, 75, 44, 56, 75, 44, 56, 75, 44, 56, 75,
       44, 56, 33, 20, 61, 15, 61, 33, 20, 61, 15, 61, 33, 20, 61, 15, 61,
       33, 20, 61, 15, 61, 33, 20, 61, 15, 61, 37, 37, 37, 37, 37, 42, 67,
       42, 67, 42, 67, 62, 67, 62, 67, 62, 67, 34, 62, 51, 39, 41, 34, 62,
       51, 39, 41, 34, 62, 51, 39, 41, 34, 62, 51, 39, 41, 34, 62, 51, 39,
       41, 34, 62, 51, 39, 41, 34, 62, 51, 39, 41, 34, 62, 51, 39, 41, 54,
       54, 42, 42, 42, 70, 70, 70, 70, 42, 42, 42, 42, 58, 77, 17, 58, 77,
       17, 58, 77, 17,  7, 39, 79,  7, 39, 79,  7, 39, 79,  7, 39, 79, 32,
       51, 42, 19, 19, 22, 22, 22, 71, 71, 71, 55, 55, 55, 55,  8,  8, 69,
       39, 79, 39, 79, 39, 79, 65, 65, 65, 45, 43, 66, 45, 43, 66, 12, 12,
       12, 12, 18, 18, 38, 38, 26, 26, 26, 26, 50, 50, 25,  5, 25,  5, 25,
        5, 24, 73, 73, 43, 12,  4,  8, 50,  4, 12, 12, 68,  4, 18, 18, 67,
       24, 18, 18, 18, 18,  8, 18, 68, 27, 27, 27], dtype=int32)
```

```
#clustering  and scatterplot for chatgpt embeddings

from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import numpy as np
```

```
# Initialize K-means with desired number of clusters
kmeans = KMeans(n_clusters=80, random_state=42)

# Fit K-means to your embedding matrix
chatgpt_cluster_labels = kmeans.fit_predict(chatgpt_embeddings)

# Generate 80 distinct colors for clusters
colors = plt.cm.tab20(np.linspace(0, 1, 80))

# Create a dictionary mapping cluster labels to colors
color_dict = {cluster_label: color for cluster_label, color in zip(range(80), col

# Plot the clusters using the first two dimensions of the embedding matrix
plt.figure(figsize=(8, 6))
for cluster_label in range(80):
    cluster_mask = chatgpt_cluster_labels == cluster_label
    plt.scatter(chatgpt_embeddings[cluster_mask, 0], chatgpt_embeddings[cluster_m
                c=[color_dict[cluster_label]], label=f'Cluster {cluster_label}')

plt.title('Visualization of Clusters for chatgpt embeddings')
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.legend()
plt.show()
```




−0.7
−0.9
−0.8
−0.7
−0.6
−0.5
Dimension 1
Cluster 16
Cluster 17
Cluster 18
Cluster 19
Cluster 20
Cluster 21
Cluster 22
Cluster 23
Cluster 24
Cluster 25
Cluster 26
Cluster 27
Cluster 28
Cluster 29
Cluster 30
Cluster 31
Cluster 32
Cluster 33
Cluster 34
Cluster 35
Cluster 36
Cluster 37
Cluster 38
Cluster 39
Cluster 40
Cluster 41
Cluster 42
Cluster 43
Cluster 44
Cluster 45
Cluster 46
Cluster 47
Cluster 48
Cluster 49
Cluster 50
Cluster 51
Cluster 52
Cluster 53
Cluster 54
Cluster 55
Cluster 56
Cluster 57
Cluster 58
Cluster 59


Cluster 60
Cluster 61
Cluster 62
Cluster 63
Cluster 64
Cluster 65
Cluster 66
Cluster 67
Cluster 68
Cluster 69
Cluster 70
Cluster 71
Cluster 72
Cluster 73
Cluster 74
Cluster 75
Cluster 76
Cluster 77
Cluster 78
Cluster 79

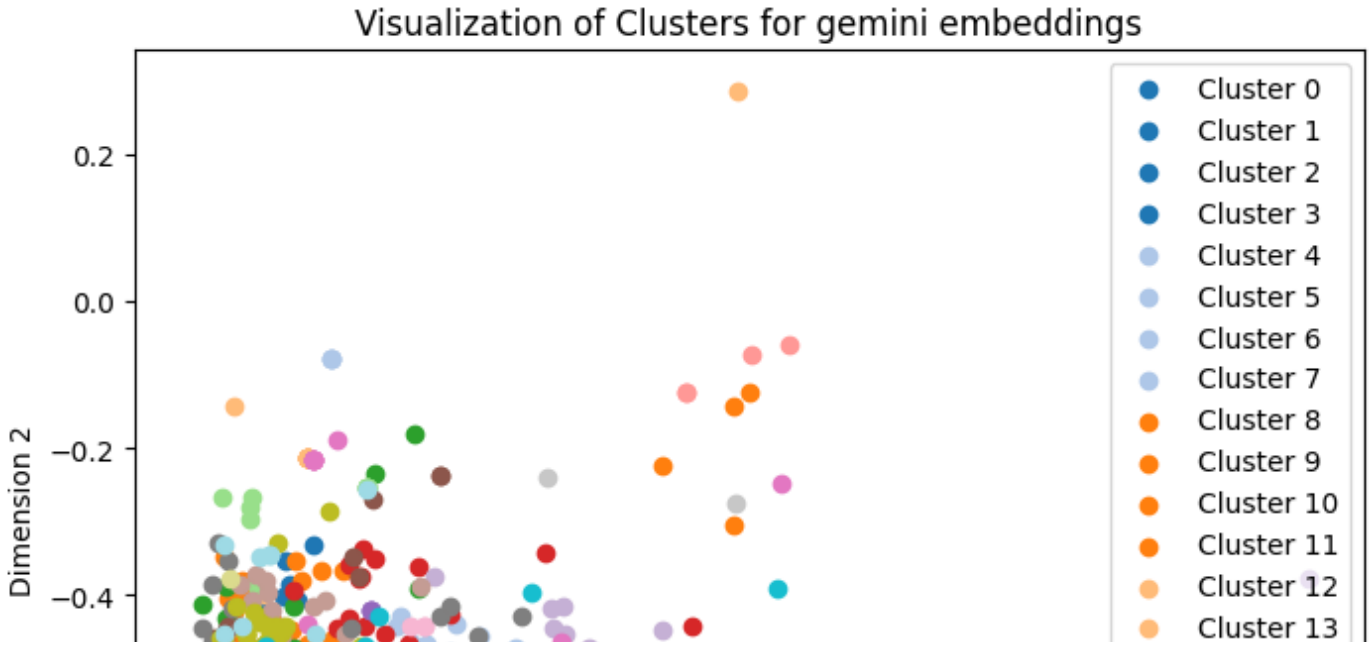
```
len(set((chatgpt_cluster_labels)))
chatgpt_cluster_labels
```

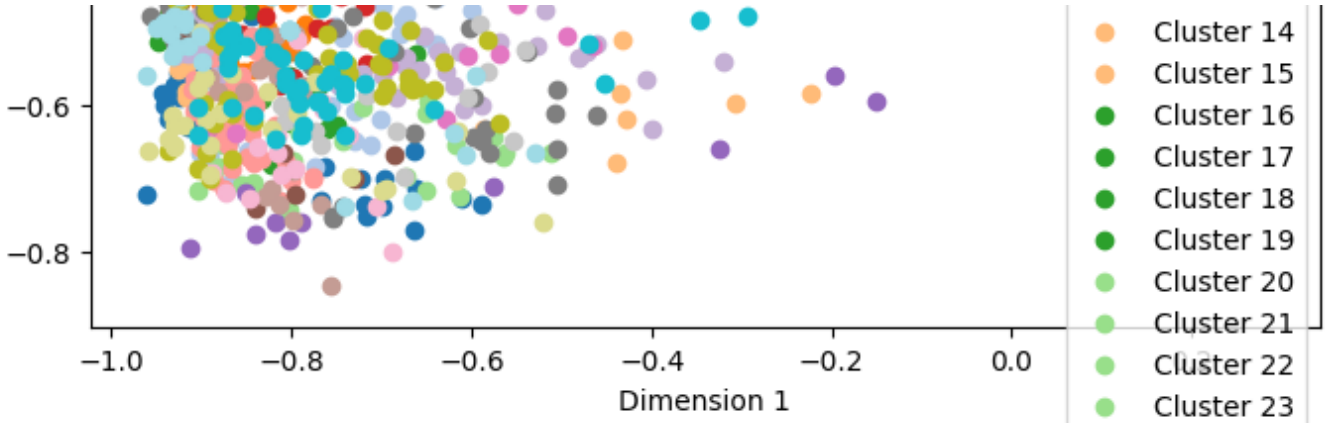
```
array([79, 46,  2,  2, 66,  2, 46, 79,  2, 46,  2, 46, 79, 79, 46, 79,  2,
        2,  2, 31, 79, 46,  2,  2, 31, 79, 46, 79, 31, 31, 79, 46, 79,  2,
        0, 22, 79,  2,  2, 46, 31, 46, 79,  2, 55, 31, 46, 55,  2, 13,  2,
       46, 79, 36, 46,  2, 46, 79,  2,  2,  6,  6, 31, 74, 39,  4, 20,  0,
       39, 38, 14, 39, 53,  4, 14, 39, 21,  4, 14, 39, 39, 39, 14, 74,  0,
       39, 14, 74, 39, 27, 14,  0, 53, 39, 38,  0, 53, 20, 20, 59,  6, 20,
        0, 17, 66, 20,  0, 17, 66,  0, 74, 20, 63, 37,  0,  6,  6,  9, 79,
       63, 48, 45, 45, 31, 48, 52,  9, 66, 66, 45, 48, 28, 48, 45, 45, 45,
       48,  1, 79, 31, 31, 66, 48, 36, 31, 31, 45, 45, 48, 28, 48, 66, 45,
       45, 48, 38, 22, 45, 66, 45, 48, 63, 54, 45, 31, 45, 11, 59, 47, 29,
       59, 36, 29, 44, 47, 29,  9,  2, 63,  1, 31, 63,  9,  2, 46,  0,  8,
       27,  9, 62, 63,  9, 62, 31,  9,  1, 27, 13, 73,  6, 28, 12, 79, 37,
       62,  9, 33, 54, 26,  9, 42, 54, 26, 52, 72, 62, 49, 79, 12, 72, 52,
       63, 12, 33, 33,  2, 72, 72, 67, 52,  8, 72, 52, 63,  8, 33, 26, 79,
       25, 11, 11, 49, 18, 39, 11, 49, 25, 56, 68, 62, 25, 38, 56,  1, 15,
       11, 11, 38, 12, 12, 75, 75, 12, 21, 60, 60, 12, 12, 75, 68,  8,  8,
       75, 75, 12, 49,  3, 56, 28, 23, 56,  3,  9, 56, 11, 17, 11, 11, 38,
       56, 74, 23, 17, 23, 37, 63, 74, 23, 52, 13, 37,  2, 21, 23, 79, 74,
       23, 22, 42, 23, 63, 30, 52, 37, 37, 16, 37,  5, 79, 13, 37, 65,  0,
       73,  2, 37, 55,  1, 51, 21,  2, 23, 55, 16, 55, 36, 79, 13, 23, 21,
        1, 49,  9, 23,  6,  1, 46,  5,  2, 59, 55, 21,  1, 72, 22, 37, 13,
       63, 26, 33,  2, 38, 38,  1, 16, 33,  2, 59, 36, 49,  1, 26,  9,  6,
       13, 26, 63, 68, 36, 23, 11, 68, 40, 12, 56, 11, 28, 54, 21, 74, 63,
       54, 78, 74, 13, 72, 78, 11, 42, 72, 67, 18,  3, 40, 70, 18, 70, 26,
       15, 26, 18, 18, 70, 74, 11, 56, 74, 11, 56, 42, 11, 60, 42, 11, 60,
       38, 56, 74, 54, 51, 42, 38, 63, 67, 11, 11, 74, 42, 56, 38, 42, 13,
       78, 37, 74, 13, 21,  6, 38, 23, 74, 59,  1, 74, 59,  6, 42, 59, 37,
       13, 59,  0, 27, 58, 13, 58,  6, 27, 58, 58, 58, 39, 27, 58, 58, 38,
        0, 39, 58, 20,  0, 39, 27, 58, 14, 39, 60, 11, 11, 11, 11,  9, 59,
       74, 74, 59, 13, 63, 37, 17, 38, 17, 37, 66, 23, 76,  7,  6, 37, 59,
       76, 41, 37, 55, 38, 66, 24,  9, 66, 59, 16, 77,  6, 55, 21, 55, 32,
        6, 43, 23, 55, 77, 66, 55, 23, 26, 77,  1, 57, 23, 21, 77, 42, 68,
       56, 42,  1, 63, 45, 21, 59, 59, 38, 38, 37, 38, 66, 17, 41, 22, 52,
       61, 22, 63, 56, 37, 33, 43, 36, 42, 36, 43, 33, 43, 43, 38, 36,  3,
       16, 55,  3, 17, 21, 79, 55, 75, 60, 60, 75, 68, 51, 60, 56, 56, 45,
       17,  9, 13, 52, 38, 21, 55, 22, 36, 70, 15,  7, 15, 15, 71,  7, 41,
        7, 41,  7,  7, 71, 64, 34, 71, 32, 34, 35, 41, 69, 69, 19, 69, 19,
       69,  7,  7, 41,  7, 69,  3, 51,  7, 60, 50, 51, 75, 68, 10,  7, 50,
       32, 68, 68, 60, 41, 51,  3, 28,  7, 41, 75], dtype=int32)
```

```
#clustering  and scatterplot for gemini embeddings
```

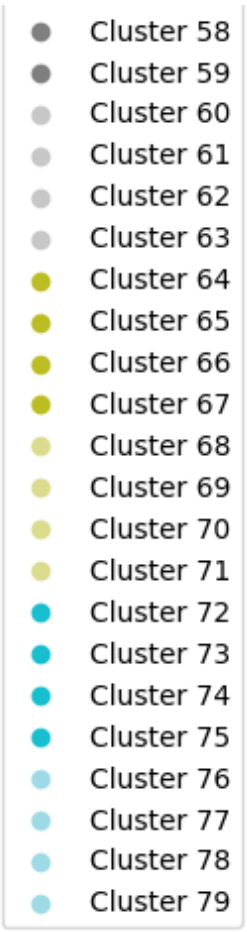
```
from sklearn.cluster import KMeans
```

```
import matplotlib.pyplot as plt
import numpy as np

# Initialize K-means with desired number of clusters
kmeans = KMeans(n_clusters=80, random_state=42)

# Fit K-means to your embedding matrix
gemini_cluster_labels = kmeans.fit_predict(gemini_embeddings)

# Generate 80 distinct colors for clusters
colors = plt.cm.tab20(np.linspace(0, 1, 80))

# Create a dictionary mapping cluster labels to colors
color_dict = {cluster_label: color for cluster_label, color in zip(range(80), col

# Plot the clusters using the first two dimensions of the embedding matrix
plt.figure(figsize=(8, 6))
for cluster_label in range(80):
    cluster_mask = gemini_cluster_labels == cluster_label
    plt.scatter(gemini_embeddings[cluster_mask, 0], gemini_embeddings[cluster_mas
                c=[color_dict[cluster_label]], label=f'Cluster {cluster_label}')

plt.title('Visualization of Clusters for gemini embeddings')
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.legend()
plt.show()
```





Cluster 14
Cluster 15
Cluster 16
Cluster 17
Cluster 18
Cluster 19
Cluster 20
Cluster 21
Cluster 22
Cluster 23
Cluster 24
Cluster 25
Cluster 26
Cluster 27
Cluster 28
Cluster 29
Cluster 30
Cluster 31
Cluster 32
Cluster 33
Cluster 34
Cluster 35
Cluster 36
Cluster 37
Cluster 38
Cluster 39
Cluster 40
Cluster 41
Cluster 42
Cluster 43
Cluster 44
Cluster 45
Cluster 46
Cluster 47
Cluster 48
Cluster 49
Cluster 50
Cluster 51
Cluster 52
Cluster 53
Cluster 54
Cluster 55
Cluster 56
Cluster 57



In the above plots 80 clusters and 661 data points were plotted with 80 different colors and we can see that the clusters are overlapping it means that datapoints belongs to more than one clusters

```
len(set((gemini_cluster_labels)))
gemini_cluster_labels
```

```
array([ 8,  8,  8,  8,  8,  8, 18, 18,  8,  8, 76,  8, 64, 24, 56,  8,  8,
       38,  8,  8,  8, 18,  8,  8,  8,  9, 18,  8, 18, 18, 18, 18,  8, 18,
       78, 11, 11, 31, 11, 11, 25, 25, 25, 25, 71, 18, 21, 21, 21, 21, 78,
       78, 25, 78, 25, 31, 31, 41, 31, 41, 72, 76, 76,  2,  2, 43, 43, 43,
       43, 43,  3, 47,  3, 72, 30, 70, 72, 71, 71, 71, 25,  6, 21, 32,  8,
       47, 43, 72, 43,  2, 43, 70, 43, 72, 14, 72, 45,  2,  2, 38, 76, 76,
       76, 38, 76, 29, 29, 38, 64,  2, 29, 72, 72,  2, 64, 29, 64, 19, 44,
       40, 34, 17, 44, 34, 52, 40, 40, 44, 52, 44,  0,  5,  0, 44, 44, 17,
       57, 14, 57, 42, 48, 57, 66, 66, 66, 66, 79, 27, 66, 66, 66, 64, 66,
       24, 27,  0, 27, 66, 34, 66,  5, 14, 14, 36,  5, 14, 57,  0, 79, 79,
       79, 79, 42, 12, 13, 27, 27, 66, 38,  3, 18, 18,  8, 55, 51, 36, 38,
       18,  9, 37, 78, 19, 57, 57, 57, 59, 57, 65,  1, 69, 57, 28, 36, 48,
       65, 57, 73, 37, 60, 37, 14, 69, 73, 30,  6, 59, 65, 65, 73, 59, 30,
        6, 38,  9, 30, 37, 65, 37,  6, 73, 22, 63, 60, 37, 65, 74, 73,  6,
       74,  6, 54, 26, 74, 26, 54, 73,  6, 65, 60, 59, 36, 65,  6,  6, 19,
       73, 60, 55, 59, 73, 30, 59,  9, 37, 73, 73, 65, 73, 19,  6,  9, 22,
       15, 30, 59, 48, 65, 65, 30, 72,  9, 73, 32, 32, 65,  4, 30,  1, 22,
       22, 72, 72, 64,  8, 22, 70, 73, 64, 70, 62,  2, 38, 26, 39, 54,  2,
        2,  2, 70, 15, 22, 47, 76, 64, 38, 64, 70, 71, 49, 76, 19, 49, 35,
       13, 75, 64, 37, 76,  8, 56, 64, 56, 56, 38, 64, 19, 15, 47, 72, 75,
       47, 14,  7, 20,  7, 20,  7, 67, 29, 56, 38, 56, 70, 75, 20, 64, 56,
       22, 47, 35, 60, 73, 28,  1, 61, 67, 49, 49, 56, 49, 49, 76, 38, 47,
       76, 76, 38,  6, 60, 63,  6, 55, 77, 77, 77,  2,  2,  2, 15,  8,  3,
        3, 43, 36, 65, 36, 65, 29, 38, 64, 24,  1, 46,  1,  4, 26,  6,  4,
       73, 19,  1, 37,  4, 28, 36,  1, 73, 23, 65,  6,  6, 30, 14, 22, 37,
       28, 30, 70, 30,  9, 30, 30, 28, 75, 30, 15, 70, 30, 72, 30, 19, 74,
       22, 29, 29, 29, 22, 57, 73, 73, 30,  9, 29, 15, 29, 30, 29, 29, 65,
       65, 65, 78, 43, 15, 64, 64, 56, 47, 74, 43, 73, 10, 56, 10, 10, 19,
       64, 64,  3, 76,  2, 70, 45, 20, 56, 76, 30,  5, 22, 30, 30, 70,  9,
        9, 29, 29, 29, 24,  9,  9,  9, 29, 29, 16, 54, 55, 73, 69, 63, 37,
       33, 54, 36, 48, 54, 48, 55, 48, 39, 48, 45, 68, 45,  9, 13, 73, 74,
       45, 13, 68, 68, 13, 68, 35, 68, 13, 35, 13,  1, 23, 46,  1,  1,  8,
       76, 38, 47, 29, 64, 23, 29,  5, 29, 24,  5, 52, 24, 24, 44, 24, 27,
        5, 44, 44, 24, 64, 39, 72, 39, 76, 37,  9,  2, 47, 15, 15, 70, 65,
       64, 29, 37, 63, 59, 37, 47, 77, 23,  1, 72, 38, 57, 15,  2, 70, 30,
       59, 59, 65, 36, 73, 65, 46,  1, 46, 66,  2,  0,  0,  0, 52, 46, 46,
       50,  4, 23, 68, 14, 68, 37, 28,  1, 22, 50, 23,  4, 63,  4, 42, 50,
       42, 55, 50, 46, 32, 60,  6, 23,  1, 23, 32, 36, 58, 37, 65, 58, 53,
       58, 23, 23, 42, 77, 65, 32, 65, 77, 23, 55], dtype=int32)
```