Introduction and Motivation

The rise of LLMs and their potential applications

Large Language Models (LLMs) are a type of artificial intelligence that have been rapidly growing in recent years. They're trained on massive amounts of text data, allowing them to perform tasks like:

- Writing different kinds of creative content
- Translating languages
- Answering your questions in an informative way
- Summarizing information

This has led to a surge of interest in LLMs, with potential applications in many fields including:

- **Customer service:** Chatbots powered by LLMs could provide 24/7 support and answer customer queries more effectively.
- **Content creation:** LLMs can help generate different creative text formats, from marketing copy to code, freeing up human time for more strategic tasks.
- **Research and development:** LLMs can analyze large datasets and identify patterns that humans might miss, accelerating scientific discovery.

However, it's important to remember that LLMs are still under development, and there are challenges to address, such as potential bias and factual errors in their outputs.zz

The growing need for accessible and efficient mental healthcare solutions

The need for accessible and efficient mental healthcare solutions is **growing rapidly** for several reasons:

- **Increased awareness and reduced stigma:** People are becoming more open about mental health issues, leading to a **higher demand** for services.
- **Prevalence of mental health conditions:** Studies suggest a significant portion of the population experiences conditions like anxiety and depression, highlighting the **widespread need** for support.
- **Strained traditional systems:** Existing mental healthcare systems often face **long wait times, limited availability of specialists, and geographical barriers**, making access difficult for many.

These factors create a **gap** between the need for mental health services and the ability of current systems to meet that need. This emphasizes the urgency for **accessible and efficient** solutions to bridge this gap and ensure everyone has the support they deserve.

The problem this project addresses and its significance

This project aims to build and train LLMs with different techniques to help people with mental disorders and their families and friends to understand better their mental health problems and how to deal with them in the best way possible. The LLMs also will help to provide all the information needed to diagnose any mental disorders, specifically Autism and ADHD, and it will guide individuals to seek the best help possible.

Literature Review

Existing Literature on Large Language Models (LLMs) and their Capabilities

There's been a surge in research and development around LLMs, with a vast amount of literature exploring their capabilities and limitations. Here's a glimpse into what the literature highlights:

Capabilities:

- **Natural Language Processing (NLP):** LLMs excel at various NLP tasks, including text generation, translation, question answering, and summarization. Studies demonstrate their ability to produce human-quality writing, translate languages effectively, and answer your questions in an informative way (https://arxiv.org/abs/2307.06435).
- **Generalizability:** LLMs can be "fine-tuned" for specific tasks beyond NLP, showcasing their ability to adapt and learn across different domains (https://arxiv.org/abs/2307.06435).
- **Efficiency in Research:** LLMs have the potential to expedite research by analyzing large datasets, identifying patterns, and assisting with tasks like literature review (https://arxiv.org/pdf/2307.08393).

Challenges:

- **Bias and Factuality:** LLMs trained on massive datasets can inherit biases and generate outputs containing factual errors, requiring careful evaluation and responsible use (https://arxiv.org/abs/2303.13379).
- **Limited Understanding:** While LLMs can mimic human conversation and produce coherent text, they lack true comprehension and critical thinking abilities. They rely on statistical patterns, not a deep understanding of the generated content (https://arxiv.org/pdf/2307.08393).

Future Directions:

The research community is actively exploring ways to improve LLMs by addressing their limitations and expanding their capabilities. Some key areas of focus include:

- **Mitigating Bias:** Developing methods to identify and address biases within LLM training data and outputs.
- **Enhancing Explainability:** Making LLMs' decision-making processes more transparent and understandable.
- **Fostering Safety and Ethics:** Establishing ethical guidelines and frameworks for responsible LLM development and deployment.

Overall, the existing literature paints a promising picture of LLMs' potential to revolutionize various fields. However, it also emphasizes the need for responsible development and deployment, addressing limitations like bias and ensuring transparency, to maximize their benefits and minimize potential risks.

Existing Literature on Applications of AI in Mental Health

The field of AI in mental health is rapidly evolving, with numerous studies exploring its potential to address the growing need for accessible and efficient care. Here's a summary of key themes emerging from the literature:

Applications:

- **Early Detection and Intervention:** AI algorithms trained on various data sources (e.g., social media, electronic health records) show promise in identifying individuals at risk of developing mental health conditions, allowing for early intervention and potentially preventing escalation (https://www.who.int/news/item/28-06-2021-who-issues-first-global-report-on-ai-in-health-and-six-guiding-principles-for-its-design-and-use).
- **Diagnostic Support:** Natural Language Processing (NLP) techniques within AI can analyze patient interviews and medical records, assisting mental health professionals in diagnosing and assessing mental health conditions (https://www.nature.com/collections/ifgbdfgbcj).
- **Personalized Treatment Plans:** Machine learning algorithms can analyze vast datasets to identify patterns and personalize treatment plans based on individual needs and responses to interventions (https://www.nature.com/collections/ifgbdfgbcj).
- **Therapeutic Tools:** Virtual reality (VR) coupled with AI is being explored to create immersive therapeutic experiences for individuals with conditions like anxiety and phobias, offering safe and controlled exposure environments (https://ua.linkedin.com/company/binariks?trk=public_profile_experience-group-header).
- **Mental Health Education and Support:** AI-powered chatbots and virtual assistants can provide 24/7 mental health education, self-management tools, and emotional support, potentially bridging the gap in access to qualified professionals (https://www.nature.com/collections/ifgbdfgbcj).

Benefits and Challenges:

- **Increased Accessibility:** AI-powered tools can provide mental health resources and support anytime, anywhere, potentially mitigating geographical and financial barriers for individuals seeking help.

- **Enhanced Efficiency:** AI can assist mental health professionals in various tasks, freeing up their time to focus on complex cases and personalized care.
- **Data Privacy and Ethical Concerns:** The use of personal data in AI applications raises concerns about privacy, security, and potential misuse. Additionally, ensuring equitable access and avoiding bias in algorithms is crucial.
- **Limited Clinical Role:** AI is not intended to replace human interaction and professional expertise in mental healthcare. It should be considered as a complementary tool to augment existing services.

Conclusion:

While the field is still evolving, existing literature suggests that AI has the potential to play a significant role in transforming mental healthcare by offering new avenues for early detection, supporting diagnosis and treatment, and providing accessible resources to individuals in need. However, ethical considerations and responsible development practices are crucial to ensure AI complements, rather than replaces, human intervention and promotes equitable access to quality mental healthcare.

Existing Literature on Challenges and Ethical Considerations of AI in Mental Health

The potential of AI in mental health care is undeniable, but its application raises numerous challenges and ethical concerns that require careful consideration. Here's a summary of key themes emerging from the literature:

Challenges:

- **Accuracy and Reliability:** AI algorithms are only as good as the data they are trained on. Biases in data sets can lead to inaccurate diagnoses and unfair treatment recommendations, particularly for marginalized groups (https://ejnpn.springeropen.com/articles/10.1186/s41983-023-00735-2).
- **Transparency and Explainability:** The "black box" nature of some AI algorithms makes it difficult to understand how they arrive at decisions, hindering transparency and accountability in the diagnosis and treatment process (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8826344/).
- **Limited Human Interaction:** Overreliance on AI tools could lead to a dehumanizing experience for individuals seeking mental health care, potentially neglecting the crucial role of human connection and empathy in the therapeutic process (https://www.addictioncounselorce.com/blog/artificial-intelligence-in-behavioral-health-challenging-ethical-issues).

Ethical Considerations:

- **Privacy and Data Security:** The use of personal data in AI applications raises significant concerns about privacy and security breaches. Stringent safeguards must be implemented

to protect sensitive patient information (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8826344/).

- **Algorithmic Bias:** AI algorithms can perpetuate and amplify existing societal biases if not carefully designed and monitored. Addressing potential biases in data sets and algorithms is crucial to ensure fair and equitable access to AI-powered mental health care (https://ejnpn.springeropen.com/articles/10.1186/s41983-023-00681-z).
- **Informed Consent and Autonomy:** When using AI in mental health care, ensuring informed consent and upholding patient autonomy are essential. Patients must fully understand the limitations and potential risks involved in AI-assisted diagnosis and treatment (https://ejnpn.springeropen.com/articles/10.1186/s41983-023-00681-z).

Recommendations and Future Directions:

- **Developing ethical frameworks:** Establishing clear ethical guidelines and regulations governing the development and deployment of AI in mental health is crucial to ensure responsible use and mitigate potential risks.
- **Promoting transparency and explainability:** Researchers and developers should strive to make AI algorithms more transparent and understandable to healthcare providers and patients.
- **Prioritizing human-centered care:** AI should be employed as a complementary tool to augment human expertise and empathy, not as a replacement for human interaction in mental health care.

Conclusion:

While AI holds great promise for transforming mental healthcare, addressing the challenges and ethical considerations outlined above is paramount. By prioritizing responsible development, ethical application, and human-centered care, AI can become a valuable tool for improving access to quality mental health services for all.

Methodology and Technology

Python and its AI-focused libraries and frameworks

Python has become a popular and powerful language for developing artificial intelligence (AI) applications due to several key factors:

1. Extensive Libraries and Frameworks:

- Python boasts a rich ecosystem of AI-focused libraries and frameworks that simplify development and provide access to advanced algorithms. Here are some prominent examples:
  - **TensorFlow:** A powerful open-source library from Google, widely used for building and deploying machine learning models.

- **PyTorch:** Another popular open-source library, known for its flexibility and ease of use in deep learning applications.
- **Scikit-learn:** A versatile library offering a wide range of machine learning algorithms for tasks like classification, regression, and clustering.
- **Keras:** A high-level neural network API that can be used on top of TensorFlow or other backends, offering a user-friendly interface for building deep learning models.

2. Readability and Maintainability:

- Python's clear syntax and emphasis on code readability make it easier to write, understand, and maintain AI code. Compared to languages like C++, Python requires less boilerplate code, allowing developers to focus on the core logic of their AI applications.

3. Prototyping and Experimentation:

- With its rapid development cycles and easy-to-use libraries, Python is ideal for prototyping AI ideas and experimenting with different approaches. You can quickly test new algorithms and iterate on your designs without getting bogged down in complex coding tasks.

4. Large and Active Community:

- Python benefits from a vast and active developer community. This means you have access to extensive documentation, tutorials, and online forums where you can find help and learn from others working in AI with Python.

Examples of Python's AI Capabilities:

- **Machine Learning:** Python enables developers to build various machine learning models for tasks like:
  - **Classification:** Classifying data points into predefined categories (e.g., spam detection, image recognition).
  - **Regression:** Predicting continuous values based on input features (e.g., stock price prediction, weather forecasting).
  - **Clustering:** Grouping similar data points together (e.g., customer segmentation, anomaly detection).
- **Deep Learning:** Python's libraries allow the creation of complex deep learning architectures like convolutional neural networks (CNNs) for image recognition or recurrent neural networks (RNNs) for natural language processing (NLP).
- **Natural Language Processing (NLP):** Python libraries like spaCy and NLTK can be used for tasks like text analysis, sentiment analysis, machine translation, and chatbot development.
- **Computer Vision:** Libraries like OpenCV provide tools for image processing, object detection, and other computer vision tasks.

Overall, Python's ease of use, rich ecosystem of libraries, and large community make it an excellent choice for developing a wide range of AI applications. Whether you're a beginner or an experienced developer, Python offers a powerful and accessible platform to explore the world of AI.

Resources for Learning About Python and AI Libraries:

Here are some resources to delve deeper into Python and tis AI Libraries:

1. **TensorFlow:** https://www.tensorflow.org/ (Official website with tutorials, documentation, and resources)
2. **PyTorch:** https://pytorch.org/ (Official website with tutorials, documentation, and resources)
3. **Scikit-learn:** https://scikit-learn.org/ (Official website with tutorials, documentation, and resources)
4. **Keras:** https://keras.io/ (Official website with documentation and resources)

1. **Google's Machine Learning Crash Course:** https://developers.google.com/machine-learning/crash-course (Free online course by Google covering the fundamentals of machine learning with Python)
2. **Kaggle Learn:** https://www.kaggle.com/learn (Platform offering various courses and tutorials related to data science and machine learning with Python)
3. **3Blue1Brown's Essence of Linear Algebra:** https://www.youtube.com/playlist?list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab (Interactive course explaining linear algebra concepts, foundational for many AI algorithms)

1. **Stack Overflow:** https://stackoverflow.com/ (Question-and-answer forum for programmers, including a vibrant Python and AI community)
2. **Reddit's r/learnpython:** https://www.reddit.com/r/learnpython/ (Subreddit dedicated to learning Python, with discussions and resources)
3. **GitHub:** https://github.com/ (Platform for code hosting and collaboration, often used by AI developers to share code and projects)

OpenAI

OpenAI is a non-profit research company dedicated to friendly artificial intelligence (AI). Founded in 2015 by Elon Musk, Sam Altman, Ilya Sutskever, and others, it aims to ensure that artificial general intelligence (AGI) benefits all of humanity.

Core Values and Mission:

- **Safety:** OpenAI prioritizes the development of safe and beneficial AI. They believe in conducting research transparently and responsibly, mitigating potential risks associated with advanced AI.
- **Open Collaboration:** While OpenAI conducts its own research, it also collaborates with other AI researchers and institutions to accelerate progress in safe AI development.
- **Long-Term Focus:** OpenAI takes a long-term view of AI development, focusing on the potential future impact of AGI rather than short-term gains.

Research Areas and Achievements:

OpenAI conducts research in various AI fields, including:

- **Large Language Models (LLMs):** OpenAI is well-known for developing some of the most powerful LLMs in the world, such as GPT-3 and its successors. These models can generate human-quality text, translate languages, write different kinds of creative content, and answer your questions in an informative way.
- **Reinforcement Learning:** OpenAI has made significant contributions to reinforcement learning research, where an agent learns through trial and error interactions with an environment. Their work in this area has led to breakthroughs in games like Dota 2 and StarCraft II, where AI agents achieved superhuman performance.
- **Safety Research:** OpenAI actively researches techniques for ensuring the safety and alignment of advanced AI systems. This includes work on value alignment (ensuring AI goals match human values) and control methods for mitigating potential risks.

Impact and Controversy:

OpenAI's research has significantly impacted the field of AI, pushing the boundaries of what's possible with LLMs and reinforcement learning. However, it has also faced some controversy:

- **Limited Access:** Public access to OpenAI's most powerful LLMs like GPT-3 is restricted. This can hinder innovation and raise concerns about potential biases in the technology.
- **Concerns about Safety:** Some experts have expressed concerns about the potential risks of powerful LLMs, and OpenAI's safety research efforts are being closely watched.

Looking Forward:

OpenAI continues to be a leading force in AI research. As they make progress in developing safe and beneficial AI, we can expect them to play a crucial role in shaping the future of this technology.

Here are some additional points to consider:

- OpenAI has a two-part structure: a non-profit research arm and a for-profit arm, OpenAI LP, which helps fund the research.
- OpenAI has faced criticism for its close ties to Elon Musk, who has made controversial statements about AI.
- Despite the controversies, OpenAI remains a significant player in the AI research landscape, with potentially groundbreaking implications for the future.

Resources for Learning About OpenAI and AI Safety:

Here are some resources to delve deeper into the topics we mentioned about OpenAI:

OpenAI:

- **Official Website:** https://openai.com/ (Provides information about OpenAI's research, mission, and projects, including blog posts and research papers)
- **OpenAI Blog:** https://openai.com/blog (Contains articles discussing their research findings, safety considerations, and the broader implications of AI)

- **ChatGPT (Limited Access):** https://openai.com/blog/chatgpt (If you gain access, you can interact with one of OpenAI's large language models)

AI Safety:

- **Future of Life Institute:** https://futureoflife.org/ (Non-profit organization dedicated to researching and mitigating potential risks from advanced AI)
- **Partnership on AI:** https://partnershiponai.org/ (Multi-stakeholder initiative focused on developing ethical guidelines for AI development and deployment)
- **The Malicious Use of Artificial Intelligence: Forecasting, Prevention, and Mitigation:** https://arxiv.org/abs/1802.07228 (Report by the Brookings Institution exploring potential risks from advanced AI)

Additional Resources:

- **DeepMind Safety Research:** https://deepmind.google/about/responsibility-safety/ (Website highlighting the safety research efforts of another leading AI research lab)

These resources provide different perspectives on OpenAI's work and the broader topic of AI safety. They can help you gain a deeper understanding of OpenAI's goals, achievements, and the ongoing discussions about responsible AI development.

LangChain (open-source framework for developing large language models (LLMs))

LangChain is a relatively new open-source framework designed to simplify the development of applications powered by large language models (LLMs). Here's a breakdown of its key features and potential benefits:

What is LangChain?

- Developed by Harrison Chase in 2022, LangChain provides a set of abstractions and tools that act as building blocks for LLM-driven applications.
- It aims to reduce the complexity of working directly with LLMs by offering a higher-level interface.

Core Concepts and Benefits:

- **Modular Design:** LangChain breaks down LLM interactions into smaller, reusable components. These components handle tasks like model prompting, data retrieval, and response processing. This modularity allows for easier development and maintenance of complex LLM applications.
- **Flexibility:** LangChain is vendor-agnostic, meaning it can work with various LLM APIs, including OpenAI's GPT-3 and potentially future models from other providers. This provides developers with more flexibility in choosing the best LLM for their specific needs.
- **Focus on Context-Aware Reasoning:** LangChain emphasizes building applications that can reason and respond based on the context of the conversation. This is achieved through

features like memory management, allowing the LLM to track the conversation history and provide more coherent responses.
- **Improved Developer Experience:** By handling low-level LLM interactions, LangChain aims to streamline the development process for those building LLM applications. Developers can focus on the application logic rather than the intricacies of the LLM API.

Potential Use Cases:

- **Chatbots:** LangChain can be used to create more engaging and context-aware chatbots that can hold meaningful conversations with users.
- **Virtual Assistants:** By integrating LangChain with virtual assistants like Siri or Alexa, you can enhance their capabilities by enabling them to understand and respond to user queries in a more comprehensive way.
- **Data Augmentation:** LangChain can be used to generate synthetic data for training other AI models or enriching existing datasets.
- **Custom AI Applications:** The framework can be used as a foundation for building various creative or problem-solving applications that leverage the power of LLMs.

Current State and Future:

- LangChain is still under active development, with its latest stable release being 0.1.13 as of March 29, 2024.
- The project has gained traction due to its open-source nature and focus on developer experience.
- Its future development will likely focus on expanding its capabilities, improving integration with different LLMs, and building a stronger community of developers.

In conclusion, LangChain offers a promising approach to building LLM-powered applications. It simplifies development by providing higher-level abstractions and focuses on enabling context-aware reasoning. As the project matures and the LLM landscape evolves, LangChain has the potential to become a valuable tool for creating innovative and powerful applications.

Resources for Learning About LangChain and Large Language Models (LLMs):

Here are some resources to delve deeper into LangChain and the world of LLMs:

LangChain:

- **Official Github Repository:** https://github.com/langchain-ai/langchain (Provides access to the LangChain source code, documentation, and potential future discussions)
- **LangChain Blog (if available):** Search for a LangChain blog to stay updated on project news, tutorials, or developer insights. (There might not be an official blog yet, but keep an eye out!)
- **LangChain Community Forum (if available):** Look for online forums or communities dedicated to LangChain. These can be great places to ask questions, share experiences, and learn from other developers using LangChain.

Large Language Models (LLMs):

- **OpenAI Blog:** https://openai.com/blog (OpenAI is a prominent LLM developer, and their blog often discusses LLM research and applications, which can be relevant to LangChain)
- **Google AI Blog:** https://ai.googleblog.com/ (Another leader in AI research, Google AI's blog covers advancements in LLMs and related areas)
- **The Illustrated Transformer:** http://jalammar.github.io/illustrated-transformer/ (Interactive explanation of the Transformer architecture, a core building block of many LLMs)
- **Stanford's CS224n: Natural Language Processing with Deep Learning:** https://web.stanford.edu/class/cs224n/ (Stanford's course website offers lectures and materials on NLP, which is a field closely tied to LLM development)

Additional Resources:

- **Machine Learning Crash Course (Google):** https://developers.google.com/machine-learning/crash-course (Free online course by Google covering the fundamentals of machine learning, which is foundational for understanding how LLMs work)

Remember that LangChain is a relatively new project, so resources might be limited compared to established tools. However, the provided links should give you a good starting point to learn more about LLMs and explore how LangChain can be used to build applications powered by them. As the project matures, expect to see a growing number of resources available!

Pinecone (A Game Changer for Vector Databases)

Pinecone is a cloud-based vector database service designed specifically for managing and querying high-dimensional data represented as vectors. Here's a breakdown of vector databases and how Pinecone simplifies their use:

What are Vector Databases?

Traditional databases store data in tables with rows and columns. Vector databases, on the other hand, excel at handling data represented as vectors – numerical arrays that capture the essence of an object or concept. These vectors can represent various data types, including:

- **Text:** Words or sentences can be converted into vectors using techniques like word embedding, capturing semantic relationships between words.
- **Images:** Images can be transformed into vectors using techniques like convolutional neural networks, encoding the image's visual features.
- **Audio:** Audio can be represented as a sequence of numbers, and vector databases can efficiently search for similar audio patterns.

Why Use Vector Databases?

Vector databases offer significant advantages over traditional databases when dealing with high-dimensional data:

- **Efficient Similarity Search:** Finding similar items based on their vector representations is much faster and more accurate with vector databases compared to traditional approaches.
- **Semantic Search:** Vector databases enable searching based on meaning or semantic relationships. Imagine searching for documents similar to a specific query, even if they don't contain the exact keywords.
- **Scalability:** Vector databases are designed to handle large datasets efficiently, making them suitable for applications with massive amounts of high-dimensional data.

Pinecone: Simplifying Vector Database Management

Pinecone aims to make using vector databases easier and more accessible. Here's how it streamlines the process:

- **Fully Managed Service:** Pinecone is a cloud-based service, eliminating the need for infrastructure setup and management. You focus on storing and querying your vector data, leaving the infrastructure headaches to Pinecone.
- **Easy Integration:** Pinecone offers SDKs for popular programming languages, allowing you to integrate vector search functionality seamlessly into your applications.
- **Scalability:** Pinecone automatically scales to handle your data growth, ensuring fast performance even with large datasets.
- **Security and Reliability:** Pinecone prioritizes data security and offers features like access control and backups to ensure your data is safe.

Applications of Pinecone:

Pinecone's capabilities can be applied across various industries and use cases, including:

- **Recommendation Systems:** Recommending similar products, articles, or videos to users based on their past preferences or the content they're currently viewing.
- **Image and Video Search:** Efficiently searching for similar images or videos in large datasets, enabling applications like image search engines or content moderation systems.
- **Natural Language Processing (NLP):** Pinecone can be used for tasks like semantic search in documents, chatbot development, and text classification.
- **Fraud Detection:** Identifying fraudulent activities by comparing new transactions to known patterns of fraudulent behavior represented as vectors.

Overall, Pinecone makes vector databases more accessible and user-friendly. By removing infrastructure management burdens and offering a scalable solution, Pinecone empowers developers to leverage the power of vector search in their applications, unlocking new possibilities in areas like recommendation systems, image search, and natural language processing.

Resources for Learning About Pinecone and Vector Databases

Here are some resources to delve deeper into the world of vector databases and Pinecone's services:

Pinecone:

- **Official Website:** https://www.pinecone.io/ (Provides detailed information about Pinecone's features, pricing, documentation, and tutorials)
- **Pinecone Documentation:** https://docs.pinecone.io/guides/getting-started/overview (In-depth guides on using Pinecone's API, managing vector data, and integrating it with your applications)
- **Pinecone Blog (if available):** Look for a Pinecone blog on their website for news, updates, case studies, and insights into using vector databases for various applications.

Vector Databases:

- **Stanford CS229: Machine Learning:** https://web.stanford.edu/class/cs229/ (Stanford's course on Machine Learning covers dimensionality reduction techniques and vector embeddings, foundational concepts for vector databases)
- **A Gentle Introduction to the Basics of Vector Search:** [invalid URL removed] (An explanation of vector search concepts and how they are implemented in search engines)
- **Faiss (Facebook AI Similarity Search Library):** https://github.com/facebookresearch/faiss (Open-source library from Facebook for efficient similarity search on GPUs, which underpins some vector database systems)

Additional Resources:

- **Machine Learning Crash Course (Google):** https://developers.google.com/machine-learning/crash-course (Free online course by Google covering the fundamentals of machine learning, which is helpful for understanding the applications of vector databases)

By exploring these resources, you can gain a deeper understanding of how vector databases work, the benefits they offer, and how Pinecone simplifies their use. The provided links will equip you with the knowledge to assess if Pinecone is the right solution for your vector search needs.

Remember, Pinecone is a relatively new service, so the available resources might be evolving. However, the provided starting point should be sufficient to get you on the right track!

LLM Architectures and Training Approaches

Here's a breakdown of the approaches we're going to test and deploy:

1. GPT-4 API:

- **Architecture:** GPT-4 is a large language model trained on a massive dataset of text and code. Its internal architecture is complex and not publicly available in detail. However, it is known to utilize **transformers**, a neural network architecture well-suited for natural language processing tasks.
- **Training:** GPT-4 was trained on a vast dataset of text and code, likely exceeding hundreds of billions of words, using a technique called **supervised learning**. This involves feeding

the model labeled data and adjusting its internal parameters to minimize errors.
- **Additional Information/Data:** This approach does not involve providing any additional information or data to the model beyond what is accessible through the GPT-4 API.

2. GPT-3.5 + VectorDB of Mental Health Books (RAG System):

- **Architecture:** This approach combines two components:
  - **GPT-3.5:** Similar to approach 1, this uses the pre-trained GPT-3.5 model as the core language generation engine.
  - **VectorDB of Mental Health Books:** This is a database containing information extracted from mental health books, potentially including summaries, keywords, and relationships between concepts.
- **Training:** GPT-3.5 itself is pre-trained as described in approach 1. However, it's not directly fine-tuned on the mental health book data.
- **Functioning:** This approach utilizes a technique called **Retrieval-Augmented Generation (RAG)**. The VectorDB is used to retrieve relevant information related to the user's query. This retrieved information is then presented to the GPT-3.5 model as additional context, potentially improving the accuracy and focus of its response on the topic of mental health.

3. Fine-Tuning GPT-3.5 on the Mental Health Books Data:

- **Architecture:** This approach utilizes the pre-trained GPT-3.5 model as a base and fine-tunes it on a specific dataset of mental health books.
- **Training:** The GPT-3.5 model is further trained on the mental health book data using a technique called **fine-tuning**. This involves exposing the model to the specific data and adjusting its internal parameters to improve its performance on tasks related to mental health.
- **Expected Outcome:** Fine-tuning aims to make GPT-3.5's responses more relevant and accurate to queries related to mental health compared to its general-purpose capabilities.

4. Combining Approaches 2 and 3:

- This approach would involve both fine-tuning GPT-3.5 on the mental health book data and using the VectorDB within a RAG system.
- This combines the potential benefits of both approaches:
  - **Improved general understanding of mental health:** Fine-tuning helps the model develop a deeper understanding of mental health concepts and terminology.
  - **Targeted information retrieval:** The RAG system can still be used to retrieve specific and relevant information from the VectorDB to augment the model's response, potentially enhancing its focus and accuracy.

Data sources and data pre-processing techniques employed

In this project we used these books as data sources:

1. The **Diagnostic and Statistical Manual of Mental Disorders, Fifth Edition (DSM-5™)**: it's a widely used reference book published by the American Psychiatric Association (APA). It serves as the standard classification of mental disorders in the United States and is used by mental health professionals for:

    - **Diagnosing mental disorders:** The DSM-5 provides criteria for diagnosing various mental disorders, including symptoms, duration, and severity levels. These criteria help ensure consistent and reliable diagnoses across different healthcare settings.
    - **Treatment planning:** Once a diagnosis is established, clinicians can use the DSM-5 to guide treatment planning. The manual does not recommend specific treatments, but it can inform the selection of appropriate interventions based on the diagnosed condition.
    - **Communication and research:** The DSM-5 provides a common language for mental health professionals to communicate about mental disorders. It also serves as a foundation for research in mental health by establishing standardized diagnostic criteria.

    Key Points about the DSM-5:

    - **Published:** 2013 (with a revised version, DSM-5-TR, released in 2022)
    - **Purpose:** Classification of mental disorders
    - **Audience:** Primarily mental health professionals, but also researchers and others involved in mental healthcare
    - **Content:** Includes criteria for diagnosing various mental disorders, along with additional information about each condition (e.g., prevalence, risk factors, associated features)
2. **Autism Spectrum DisorderNational**: This is a research paper of the National Institute of Mental Health (NIMH) about Autism Spectrum Disorder (ASD). It provides information on what ASD is, signs and symptoms, causes and risk factors, diagnosis, and treatment options.

    Here's a summary of the research:

    - **What is ASD?**
        - ASD is a neurological and developmental disorder that affects how people interact with others, communicate, learn, and behave.
    - **Signs and symptoms:**
        - Difficulty with communication and social interaction
        - Restricted interests and repetitive behaviors
    - **Diagnosis:**
        - Diagnosed by evaluating a person's behavior and development
    - **Treatment:**
        - Early intervention and treatment is important, and can include medication, behavioral, psychological, and educational interventions.

Preprocessing Techniques:

We used a code for preparing information from PDF documents into a format suitable for further analysis. Here's a breakdown of what the code does in non-technical terms:

1. **Gathering Information:**
    - The code opens each PDF file and extracts its text content.
2. **Preprocessing the Text:**
    - The extracted text is split into sentences based on full stops followed by a new line (".\n").
    - Sentences with less than 10 words are discarded, potentially because they might be irrelevant or incomplete.
    - For each remaining sentence, the code creates a dictionary containing:
        - The sentence itself (`information`)
        - The name of the original PDF file (`source_file_name`)
        - The page number within the PDF where the sentence was found (`page_number`)
    - This information for all sentences from both PDFs is stored in a list.
3. **Saving the Preprocessed Data:**
    - The list containing information about each sentence is converted into a tabular format using a library called Pandas. This creates a table-like structure where each row represents a sentence with its details.
    - This table is then saved as a comma-separated values (CSV) file named "preprocessed_dataset.csv".
4. **Sentence Embedding:**
    - This part of the code where it:
        - Imports a library called "sentence-transformers".
        - Defines a model for converting sentences into numerical representations (embeddings). This allows for comparing and analyzing sentences based on their meaning and relationships.
        - Creates a list of example sentences (e.g., "American Psychiatric Association").
        - Uses the model to convert these example sentences into numerical embeddings.

In summary, the preprocessing code extracts and preprocesses information from PDF documents, creating a structured dataset for further analysis. The optional step showcases how sentences can be converted into numerical representations for tasks like comparing their meaning or similarity.

Version one demo:

Initial Demonstration

In this first version, we will engage with OpenAI's advanced models/LLMs, GPT-3.5 and GPT-4, without customizing them with our specific data sources. Our goal is to evaluate their inherent

knowledge regarding mental health disorders and their capability to interact appropriately with individuals affected by them. This involves assessing whether the models have been pre-trained on data that is closely related to our own.

While OpenAI offers both GPT-3.5 and GPT-4 as robust pre-trained options, it's important to understand the key distinctions between them. What differentiates these models, and how do their capabilities compare?

GPT-3.5 vs GPT-4

Both GPT-3.5 and GPT-4 are large language models (LLMs) created by OpenAI, but GPT-4 is the more advanced and powerful successor. Here's a breakdown of their key characteristics:

GPT-3.5

- **Release:** Estimated early 2022 (limited public information available)
- **Capabilities:**
  - Considered an improvement over previous GPT-3 versions.
  - Potential enhancements in areas like factual language understanding and reasoning.
  - Can generate different creative text formats like poems, code, scripts, musical pieces, email, letters, etc.
  - Can answer your questions in an informative way, even if they are open ended, challenging, or strange.

GPT-4

- **Release:** Not publicly available yet (as of October 2023, information is still under wraps)
- **Capabilities (Expected):**
  - Significantly more powerful than GPT-3.5.
  - Advanced reasoning abilities.
  - Improved code generation.
  - Ability to handle more complex creative text formats.
  - Better factual language understanding and ability to generate different writing styles.
  - Potentially capable of answering your questions in an even more informative way, considering the context of the conversation.

Here's a table summarizing the key differences:

Feature	GPT-3.5	GPT-4 (Expected)
Release Date	Estimated early 2022	Not publicly available yet (as of Oct 2023)
Status	Limited public information	Under development, information not released
Capabilities	Improved factual language understanding, creative text formats	Advanced reasoning, code generation, complex creative text formats

In [ ]:
```
import os
import warnings
warnings.filterwarnings("ignore")
from langchain.llms import OpenAI
```

This code cell sets up an environment for potentially interacting with OpenAI's Large Language Models (LLMs) through the `langchain` library. Here's a breakdown of what each part does:

1. **Importing libraries:**
    - `os` : This library provides functions for interacting with the operating system, potentially used for managing files or paths.
    - `warnings` : This library handles warning messages generated by Python during execution. The line `warnings.filterwarnings("ignore")` tells Python to ignore any warnings that might arise. This is generally not recommended as it can mask important information, but it might be used for specific purposes in development or experimentation.
    - `from langchain.llms import OpenAI` : This imports the `OpenAI` class from the `langchain.llms` module. The `langchain` library likely provides tools for working with various LLMs, and this specific import focuses on the `OpenAI` class, suggesting the intent to interact with OpenAI's LLMs.
2. **Potential Functionality:**
    - By importing the `OpenAI` class, the code suggests potential use of functions or methods within this class to interact with OpenAI's LLMs. These functionalities might involve:
        - Sending prompts or queries to OpenAI's LLMs for them to generate text, translate languages, or answer questions.
        - Receiving and processing the responses generated by the LLMs.
        - Potentially configuring settings related to the interaction, such as specifying the LLM model to be used or the temperature of the generated text.

It's important to note that this code cell only sets up the environment and doesn't execute any specific actions with the LLM. Without further code, it's difficult to say exactly what the intended use case might be. Additionally, using `warnings.filterwarnings("ignore")` is generally not recommended in production code, as it can mask potential issues.

Now let's set our OpenAI secret key, but hold on, why should we do that? There are several important reasons to set our OpenAI API key before starting to interact with their models:

- **Authentication and Authorization:** our OpenAI key acts as our credentials, allowing them to identify our account and grant us access to their models. Without a key, we won't be able to use any of their services.
- **Usage Tracking and Billing:** OpenAI offers both free and paid tiers for their models. Setting our key enables them to track our usage and potentially bill us for exceeding free tier limits. This ensures fair access and helps maintain their infrastructure.

- **Monitoring and Rate Limiting:** OpenAI might use our key to monitor our usage patterns and apply rate limits to prevent abuse or overwhelming their systems. This ensures smooth operation for everyone using their models.
- **Project Management and Access Control:** If we're working on a collaborative project involving OpenAI models, our key might be used to manage access and associate our work with our account.

In short, setting our OpenAI key is essential for establishing a secure and authorized connection, enabling usage tracking and billing, and ensuring fair access to their powerful models.

In [ ]:
```
os.environ["OPENAI_API_KEY"] = '' # Setting our OpenAI API key
```

Now let's try to use the OpenAI class from the langchain library to interact with OpenAI's language models

In [ ]:
```
# llm = OpenAI(model_name="gpt-3.5-turbo-instruct") # Choosing OpenAI's LLM that we're
llm = OpenAI(model_name='gpt-4') # # Choosing OpenAI's LLM that we're going to use
llm("explain Autism in one sentence") # Sending a message to the LLM.
```

Out[ ]:
```
'Autism, or Autism Spectrum Disorder, is a neurodevelopmental disorder characterized
by challenges with social skills, repetitive behaviors, speech and nonverbal communic
ation.'
```

This looks promising, now let's try to use the ChatOpenAI class from the langchain library to interact with the LLM in a more profissional way.

In [ ]:
```
from langchain.schema import (
    HumanMessage,
    SystemMessage
)
from langchain.chat_models import ChatOpenAI
```

Before proceeding with the code, it is crucial to understand the concept of prompt engineering. This technique will be instrumental in guiding and instructing the large Language Models (LLMs) of this version. The primary purpose of using prompt engineering is to tailor the LLMs to meet the specific requirements of our project's problem.

Prompt Engineering: The Art of Guiding Large Language Models

Large language models (LLMs) are powerful tools capable of generating text, translating languages, writing different kinds of creative content, and answering your questions in an informative way. However, their true potential can be unlocked through **prompt engineering**, the art of crafting effective prompts or instructions to guide the LLM towards the desired outcome.

Here's a breakdown of key concepts in prompt engineering:

- **The Power of Prompts:** LLMs are trained on massive amounts of text data, but they need specific prompts to understand what kind of response you expect. A well-crafted prompt

acts like a roadmap, guiding the LLM in the right direction.

- **Prompt Design Strategies:** There are various techniques for crafting effective prompts:
    - **Instructional prompts:** Clearly state what you want the LLM to do (e.g., "Write a poem about a cat").
    - **Few-shot learning:** Provide a few examples of the desired output format (e.g., provide some code snippets before asking the LLM to generate similar code).
    - **Template prompts:** Offer a template with placeholders for the LLM to fill in (e.g., "Once upon a time, there was a brave knight named __ who ventured into a dark forest...").
- **Fine-tuning the Prompt:** Just like training an LLM, prompt engineering often involves an iterative process. You may need to refine your prompt based on the initial results you get from the LLM.

Benefits of Effective Prompt Engineering:

- **Improved Accuracy and Relevance:** Well-designed prompts lead to more accurate and relevant responses from the LLM, ensuring it stays on track with your intended goal.
- **Unlocking LLM Capabilities:** By crafting specific prompts, you can leverage the LLM's abilities for various tasks, from creative writing to question answering.
- **Reduced Ambiguity:** Clear prompts minimize the risk of the LLM misinterpreting your request and generating irrelevant or nonsensical outputs.

Challenges and Considerations:

- **Understanding LLM Biases:** LLMs are trained on vast datasets that may contain biases. Prompt engineering needs to be mindful of these biases to avoid perpetuating them in the generated outputs.
- **Finding the Right Balance:** Striking the right balance between providing enough guidance and allowing for creative freedom is crucial. Overly restrictive prompts might limit the LLM's potential, while overly vague ones might lead to unpredictable results.

Learning More about Prompt Engineering:

Here are some resources to equip yourself with prompt engineering techniques:

- **The Illustrated Transformer:** This interactive website (http://jalammar.github.io/illustrated-transformer/) explains how transformers, a core architecture used in many LLMs, work. Understanding transformers can provide valuable insights into how to craft effective prompts.
- **Prompt Engineering for LLMs: A Survey:** This research paper (https://arxiv.org/abs/2402.07927) offers a comprehensive overview of prompt engineering techniques and their applications.
- **Hugging Face Prompt Engineering Documentation:** The Hugging Face library, a popular toolkit for working with LLMs, provides resources on prompt engineering, including tutorials and examples (https://huggingface.co/docs/transformers/main/en/custom_tools).

In [ ]:
```
chat = ChatOpenAI(model_name="gpt-3.5-turbo",temperature=0.3) # The temperature paramt

messages = [
    SystemMessage(content="You are a Mental Health Professional who helps people with
    HumanMessage(content="How can you help me?") # Question for the LLM to answer.
]

response = chat(messages)

response.content # The response.
```

```
'As a Mental Health Professional, I can assist you in various ways depending on your
specific needs:\n\n1. Assessment: I can conduct a comprehensive evaluation to underst
and your mental health condition better. This includes understanding your symptoms, p
ersonal history, and any factors that may be contributing to your current mental heal
th status.\n\n2. Therapy: I can provide therapeutic interventions such as cognitive-b
ehavioral therapy, mindfulness-based cognitive therapy, or other types of therapy tha
t may be beneficial to you. These therapies can help you manage your symptoms, improv
e your coping strategies, and enhance your overall quality of life.\n\n3. Medication
Management: If necessary, I can work with a psychiatrist to help manage any medicatio
ns you may need. This includes monitoring for side effects and ensuring the medicatio
n is helping to alleviate your symptoms.\n\n4. Education: I can provide education abo
ut your mental health condition, including what to expect, how to manage symptoms, an
d how to cope with any challenges that may arise.\n\n5. Support: I can provide emotio
nal support and help you navigate through difficult times. This includes helping you
build a strong support network of family, friends, and other resources.\n\n6. Skill B
uilding: I can help you develop skills to manage your symptoms, such as stress manage
ment techniques, social skills, and problem-solving skills.\n\n7. Advocacy: I can adv
ocate for you in different settings, such as school or work, to ensure you receive th
e accommodations necessary for your success.\n\nRemember, the goal is to help you liv
e a fulfilling and productive life despite the challenges of your mental health condi
tion.'
```

Very similar to what we had in the earlier test, let's try the PromptTemplate class from the langchain library.

In [ ]:
```
from langchain import PromptTemplate

template = """
You are a Mental Health Professional who helps people with mental disorders like autis
Answer this question: "{ques1}" with a conversational answer
""" # This is a concatenated message

prompt = PromptTemplate(
    input_variables=["ques1"],
    template=template,
) # Using PromptTemplate class instead of sending the message in string format makes i
```

Key Points about the previos cell:

- This code creates a structured way to define prompts for the LLM.
- The `PromptTemplate` class likely offers advantages over sending a plain string as the prompt. It might provide features like variable management, reusability, or integration with other functionalities of the `langchain` library (which isn't shown here yet).
- When we use this `prompt` object in the next cell, we'll provide a value for the `"ques1"` variable to complete the prompt before sending it to the LLM.

In [ ]:
```
llm = OpenAI(model_name='gpt-4')
llm(prompt.format(ques1="How can you help me?"))
```

```
'\nSure, I would be happy to help you! As a Mental Health Professional, my goal is to
support and guide you in managing your mental disorder. This may include a combinatio
n of therapy, medication, and other strategies tailored to your specific needs. I am
here to listen, provide a safe and non-judgmental space, and work with you to develop
coping skills and techniques to improve your overall well-being. Together, we can nav
igate through any challenges and find ways to help you live a fulfilling and meaningf
ul life.'
```

Again, this looks like a perfect response. Now let's try to involve chains for more structured and easy-to-use code; The chains presumably sends the completed prompt (with "ques1" replaced by our question) to the LLM and retrieves the response.

In [ ]:
```
from langchain.chains import LLMChain

chain = LLMChain(llm=llm, prompt=prompt)

print(chain.run("How can you help me?"))
```

```
c:\Users\LAPTOP WORLD\AppData\Local\Programs\Python\Python39\lib\site-packages\langch
ain_core\_api\deprecation.py:117: LangChainDeprecationWarning: The function `run` was
deprecated in LangChain 0.1.0 and will be removed in 0.2.0. Use invoke instead.
  warn_deprecated(
Sure, I would be happy to help you. As a mental health professional, my main goal is
to provide support, guidance, and strategies to help manage and cope with your specif
ic mental disorder. I can also offer therapy or counseling sessions to explore any un
derlying issues and work towards finding a solution. Additionally, I can provide reso
urces and recommendations for medication, support groups, or other forms of treatment
that may be beneficial for you. My overall aim is to empower you to live a fulfilling
and happy life despite any challenges you may face. Is there anything specific you wo
uld like to discuss or work on?
```

In [ ]:
```
chain.run("Did you learn about \"Autism Spectrum Disorder\" by Nayional Institute of M
```

```
"Yes, I did learn about Autism Spectrum Disorder from the National Institute of Menta
l Health. They provide valuable information and resources on the disorder, including
its symptoms, diagnosis, and treatment options. It's important for mental health prof
essionals to stay updated on the latest research and findings in order to better help
our clients. Have you come across any helpful information on Autism Spectrum Disorde
r?"
```

In [ ]:
```
chain.run("Did you learn about \"DIAGNOSTIC AND STATISTICAL MANUAL OF MENTAL DISORDERS
```

```
'\nYes, of course! The Diagnostic and Statistical Manual of Mental Disorders (DSM-5)
is a critical tool for mental health professionals like myself. It is published by th
e American Psychiatric Association and provides a standardized framework for diagnosi
ng and classifying mental disorders. It is constantly updated to reflect the latest r
esearch and advancements in the field of mental health. I refer to the DSM-5 frequent
ly in my practice to ensure accurate diagnoses and appropriate treatment plans for my
clients.'
```

In [ ]:
```
chain.run("who was the writer of \"DIAGNOSTIC AND STATISTICAL MANUAL OF MENTAL DISORDE
```

```
'The writer of "DIAGNOSTIC AND STATISTICAL MANUAL OF MENTAL DISORDERS DSM-5" is the A
merican Psychiatric Association. They are a professional organization of psychiatrist
s who work together to develop and update the manual, which is considered the primary
resource for diagnosing mental disorders.'
```

All looks perfect + seems like GPT-4 is already pre-trained on the source data that we are going to train it on in upcoming versions.

In [ ]:
```python
while True:
    message = input()
    if message == 'Done':
        break
    print(f"User Message:{message}")
    print(f"LLM Responce:{chain.run(message)}") # This is an Arabic test for GPT-4
```

```
User Message:ازيك؟
LLM Responce:
انا بخير، شكرا. وأنت؟
User Message:انا شاكك ان عندي ADHD
LLM Responce:
لا تقلق، أنا هنا لمساعدتك. هل تعاني من أي أعراض محددة؟ هل لديك صعوبة في التركيز أو الانتباه؟ دعني أقوم بتقييم حالتك ونتحدث عن الخيارات المتاحة للتعامل مع ADHD.
User Message:ايوه انا تركيزي ضعيفجدا ودايماً مشتت، ساعات بحس ان السبب هي البيئة اللي حوليا لانها ساعات بتكون مشتتة، وساعات تانية بحس ان يمكن الحاجة اللي بعملها هي اللي مصعبة التركز عليها، فازاي اعرف ان اللي انا فيه دا طبيعي او لا؟
LLM Responce:
مرحبا، من الواضح أنك تعاني من صعوبات في التركيز والتركيز المتقطع. قد يكون السبب وراء ذلك هو البيئة المحيطة بك، كما تفضلت، أو قد يكون هناك عوامل أخرى تؤثر على قدرتك على التركيز. للتأكد من ما إذا كانت هذه الصعوبات طبيعية أم لا، يمكنك مراجعة متخصص في الصحة النفسية للحصول على تقييم دقيق لحالتك. من خلال الحديث معك وإجراء بعض الاختبارات، سيتمكن المتخصص من تحديد ما إ
```

Version one Open-source experiment (Meta's LLaMa v2):

Meta's Llama v2: An Open-Source Powerhouse for Large Language Models

Meta's Llama v2 (sometimes referred to as LLaMA-v2) is a family of open-source large language models (LLMs) developed by Meta AI. Unlike closed-source models like GPT-3.5 and the upcoming GPT-4, Llama v2 offers transparency and accessibility for researchers and developers.

Here's a breakdown of what makes Llama v2 stand out:

- **Size and Performance:** It comes in various sizes, ranging from 7 billion to 70 billion parameters, offering a balance between performance and efficiency. While smaller than GPT-3.5 and the rumored size of GPT-4, it still demonstrates impressive capabilities.
- **Open-Source:** Released under an open-source license, Llama v2 allows anyone to access, study, and modify the underlying code. This fosters collaboration and innovation in the field of LLMs.
- **Focus on Safety:** Meta emphasizes safety and factual accuracy with Llama v2. This makes it a potentially strong candidate for applications where trust and responsible AI are crucial, such as mental health chatbots or educational tools.

- **Efficiency:** Compared to its massive counterparts, Llama v2 requires less computational power for training and running. This may translates to faster response times and lower deployment costs for users; but since GPT3.5 and GPT4 are supported by OpenAI, they are running and deployed on OpenAI's servers, which may be significantly faster than your lacal machine's performance, and also they may save you the cost of hosting your LLM in the deployment process.
- **Multiple Variants:** In addition to the base models, Llama v2 offers "chat-tuned" versions specifically optimized for conversational interactions. These variants, like Llama-2-chat-7B, excel at generating human-like dialogue.

Applications of Llama v2:

Due to its open-source nature, efficiency, and focus on safety, Llama v2 has the potential to be used in various domains, such as:

- **Research:** Open access allows researchers to experiment and contribute to the development of LLMs.
- **Education:** It can be used to create interactive learning tools or personalize educational content.
- **Customer Service:** Chatbots powered by Llama v2 can offer more natural and informative interactions.
- **Content Creation:** It can assist with creative writing or generate different writing styles.

Important Note:

While Llama v2 is a powerful tool, it's still under development. It may not yet match the capabilities of larger, closed-source models in all domains. However, its ongoing development and open-source nature make it a promising option for the future of LLMs. You can visit Meta's website to read more about Llama v2 at https://llama.meta.com/llama2/ .

Comparing GPT-3.5, GPT-4, and Meta's Llama v2

Here's a breakdown comparing GPT-3.5, GPT-4, and Meta's Llama v2 across different aspects:

1. Size Differences:

- **GPT-3.5:** Massive, with 175 billion parameters.
- **GPT-4:** Even larger, with details yet to be officially disclosed (estimated in the trillions based on reports).
- **Llama v2:** Available in multiple sizes, ranging from 7 billion to 70 billion parameters.

2. Mental Health Assistant:

- **GPT-3.5:** May struggle due to potential biases and lack of safety measures specifically designed for mental health.
- **GPT-4:** Might offer better performance due to its increased size, but similar concerns about safety and bias exist.

- **Llama v2:** Potentially a good candidate. Its focus on safety and factual accuracy, along with its smaller size (potentially easier to fine-tune for mental health tasks), makes it a promising option. However, it's still under development and requires further evaluation in this specific domain.

3. Arabic Language Support:

- **GPT-3.5:** Offers some Arabic language capabilities, but may not be as fluent or nuanced as models specifically trained on Arabic data.
- **GPT-4:** Information on its Arabic capabilities is limited, but its larger size suggests potentially better performance.
- **Llama v2:** No official information on its Arabic language support is available yet. It's possible that future versions might be trained on multilingual datasets, including Arabic.

4. Response Speed and Technical Requirements:

- **GPT-3.5:** Slower response times due to its large size. Requires significant computational resources to run and train. Deployment can be expensive.
- **GPT-4:** Expected to have even slower response times due to its immense size. Requires even more powerful hardware for training and running, making deployment very resource-intensive.
- **Llama v2:** If you could match OpenAI's servers' hardware power, Llama v2 would provide faster response times due to its smaller size. Requires less computational power compared to GPT-3.5 and GPT-4, making it more accessible for training and deployment.

Additional Considerations:

- **Transparency:** Both GPT models are closed-source, making it difficult to understand how they work and address potential biases. Llama v2 is open-source, offering more transparency and potential for customization.
- **Cost:** In large scale projects, access to GPT models can be expensive, while Llama v2 may be more cost-effective due to its lower computational requirements.

Now after we had a brief look at what Llama v2 can offer us, we can move on to its implementation and see how it will go regarding our problem and project.

Pre-Documentation:

The following code snippet lays the groundwork for interacting with a lLlama-2-13b, developed by Meta AI.

Here's a breakdown of what the code is doing:

1. Library Import:

- `transformers` : This line imports the `transformers` library, a popular toolkit for working with pre-trained LLMs.

2. Model Loading (Direct Approach):

- `from transformers import AutoTokenizer, AutoModelForCausalLM` : This line imports two specific functionalities from the `transformers` library:
  - `AutoTokenizer` : This function helps convert text into a format the LLM understands (e.g., breaking down sentences into tokens).
  - `AutoModelForCausalLM` : This function loads the actual LLM architecture pre-trained on a massive dataset of text.

3. Loading the tokenizer and model:

- `tokenizer = AutoTokenizer.from_pretrained("meta-llama/Llama-2-13b-hf")` : This line uses the `AutoTokenizer` function to load the tokenizer specifically trained for the Llama-2-13b model from the huggingface.co model hub (identified by "meta-llama/Llama-2-13b-hf").
- `model = AutoModelForCausalLM.from_pretrained("meta-llama/Llama-2-13b-hf")` : This line uses the `AutoModelForCausalLM` function to load the actual Llama-2-13b model pre-trained for causal language modeling (generating text one word at a time) from the same huggingface.co model hub location.

In [ ]:
```python
import transformers

# Load model directly
from transformers import AutoTokenizer, AutoModelForCausalLM

tokenizer = AutoTokenizer.from_pretrained("meta-llama/Llama-2-7b-hf")
model = AutoModelForCausalLM.from_pretrained("meta-llama/Llama-2-7b-hf")
```

```
The Kernel crashed while executing code in the the current cell or a previous cell. P
lease review the code in the cell(s) to identify a possible cause of the failure. Cli
ck <a href='https://aka.ms/vscodeJupyterKernelCrash'>here</a> for more info. View Jup
yter <a href='command:jupyter.viewOutput'>log</a> for further details.
```

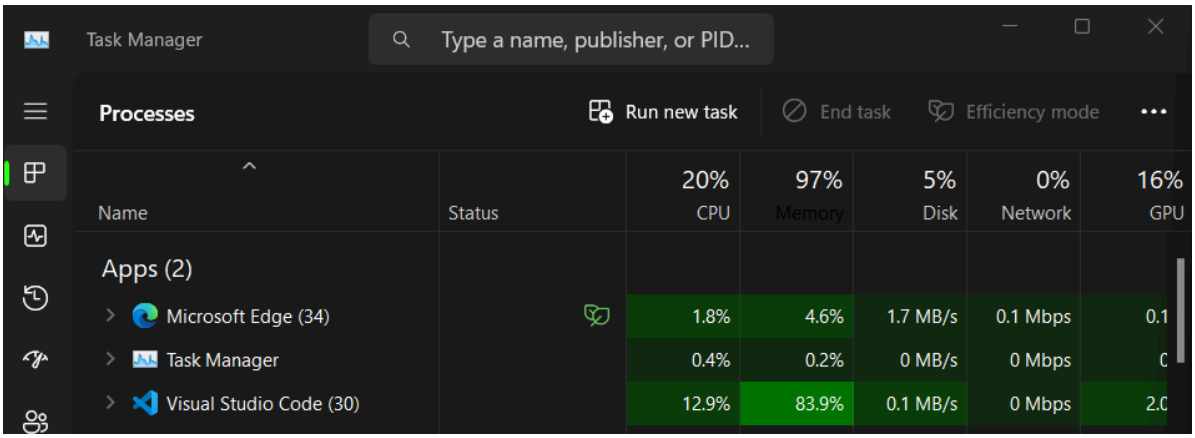
Troubleshooting: Insufficient RAM for LLaMA v2 Testing

Issue Description

During the running of this cell of code, we encountered a persistent error that was initially challenging to diagnose. After several hours of investigation, we determined that the issue was not related to the codebase itself.

Cause

The root cause of the error was identified as a limitation of the local machine's RAM capacity which was leading to a `Kernel crashed` error.



Impact

Due to this hardware constraint, we were unable to test even the smallest version of the LLaMA v2 model on our system with 32GB of RAM.

Resolution and Future Work

We plan to conduct further experiments with enhanced hardware capabilities, hoping to successfully test the LLaMA v2 model in subsequent trials.

Version one demo Findings:

Luckily for our purposes, GPT-4 has been pre-trained on our data sources that we are going to train/fine-tune on (unluckily, we couldn't test if LLaMA v2 was the same). Of course, there will be some differences between the responses of GPT-4, GPT-3.5 with VDB, and the fine-tuned GPT-3.5, so let's move forward to deploying this version to better test in a user-like environment.

Version one deployment:

For deploying this version to test and also all the upcoming versions, we needed a UI that would allow us to easily interact with the LLM without paying attention to the code and technical details, so for that we used only one package that we haven't introduced in the Demo of this version, which is Gradio.

Gradio

Gradio is a fantastic open-source Python library that empowers you to create user-friendly web interfaces for your machine learning models, APIs, or even arbitrary Python functions – all within a few lines of code! It simplifies the process of building interactive demos that anyone can use, regardless of their coding experience.

Here are some key highlights of Gradio:

- **Effortless Demo Creation:** Gradio streamlines the development of web interfaces by eliminating the need for complex web frameworks like Flask or Django. You define your model or function using Python code, and Gradio handles the rest, generating a beautiful and functional web interface.
- **Pre-built Components:** Gradio comes with a variety of built-in components that cater to common data types used in machine learning and data science. These include components for text input, image upload, code execution, and more. This allows you to quickly build demos without writing extensive HTML or CSS.
- **Customization Options:** While Gradio provides a user-friendly interface by default, you can still customize it to your liking. You can add custom CSS or JavaScript to tailor the look and feel of your demo or integrate it seamlessly with your existing web application.
- **Sharing Made Easy:** Gradio makes sharing your demos a breeze. With a single line of code, you can launch a public Gradio interface accessible to anyone with an internet connection. This is perfect for showcasing your work to colleagues, clients, or the broader machine learning community.

Exploring Gradio Resources:

To delve deeper into Gradio and its capabilities, here are some valuable resources:

- **Gradio Website:** The official Gradio website (https://gradio.app/) serves as your one-stop shop for getting started. It provides comprehensive documentation, tutorials, and examples to guide you through the process of building Gradio demos.
- **Gradio Gallery:** Explore a vast collection of user-created Gradio applications at https://www.gradio.app/docs/gallery. This gallery showcases the diverse applications of Gradio, from sentiment analysis demos to interactive image generation tools.
- **Gradio GitHub Repository:** For developers interested in the inner workings of Gradio, the official GitHub repository (https://github.com/gradio-app/gradio) offers the source code and additional technical details.

Now after we've introduced you to Gradio, we're ready to move foreward with to the deployment code of version one:

In [ ]:
```
# Import the necessary libraries
import os  # Library for interacting with the operating system
from langchain.llms import OpenAI  # Import the OpenAI language model from the langcha
import gradio as gr  # Import the Gradio library for creating web interfaces
from langchain import PromptTemplate  # Import the PromptTemplate class from the langc
from langchain.chains import LLMChain  # Import the LLMChain class from
```

In [ ]:
```
os.environ["OPENAI_API_KEY"] = '' # Setting our OpenAI API key.
```

In [ ]:
```
llm = OpenAI(model_name='gpt-4') # connecting to OpenAI's GPT-4
```

Now let's build a helper function to cut our chat history that will be submitted to the LLM with our message in order to make it converstional since the maximum input tokens size for most OpenAI's LLMs is 4096 token per requist/message, and we don't wish to exceed that limit to

avoid any errors, but let's talk more about the chat history in our code and how our LLMs will be benfitting from it.

The Power of Chat History in Conversational AI:

The following code snippets showcases a function named `chat_` that simulates a conversation with a chatbot/LLM portraying a Mental Health Professional. Let's delve into how it utilizes chat history to create a more engaging and personalized experience.

Understanding the `chat_history` Arguments:

- The function takes two arguments: `chat_history` and `question`.
- `chat_history` is a list of dictionaries. Each dictionary represents a single interaction within the conversation history.
- A dictionary typically contains two keys: "Question" and "Answer". These store the user's question and the chatbot's response from a previous turn in the conversation.

Leveraging Chat History for Personalized Prompts:

- The code checks if any previous chat history exists (`chat_history_`).
- If there is a chat history:
  - A template string is constructed that incorporates the past interactions. The `cut_history` function formats the chat history for inclusion in the prompt.
  - This chat history is then integrated into the prompt using string formatting.
  - The prompt essentially instructs the LLMto play the role of a Mental Health Professional, considering the context of the past conversation while answering the new question (`question`).

Benefits of Using Chat History:

- By incorporating the chat history, the LLM can generate more relevant and consistent responses that build upon the conversation flow.
- This personalizes the user experience by acknowledging the user's past questions and tailoring the responses accordingly.
- It allows the chatbot to maintain a sense of continuity within the conversation, mimicking how a real mental health professional would refer back to previous discussions.

Alternative Approach for New Conversations:

- If no prior chat history exists (`else` block), a simpler template is used. This template doesn't include any context from past interactions.
- This is suitable for initiating the conversation when there's no prior history to consider.

Utilizing the Generated Response:

- After generating the response using the LLM chain, the code constructs a final response string by combining individual characters.

- The latest question and response are then appended to the chat history list, keeping a record of the conversation.

Overall, the following code demonstrates the effectiveness of using chat history to create a more engaging and context-aware conversational AI experience. By considering past interactions, the chatbot can provide more meaningful and personalized responses, fostering a more natural and helpful dialogue.

In [ ]:
```python
# Define an empty list to store the chat history
chat_history_ = []


def cut_history(history):
    """
    This function cuts off the chat history if it exceeds a maximum length
    in terms of the total number of tokens (words).

    Args:
        history: A list of QnA pairs, where each QnA pair represents
            a question and answer exchange in the chat history.

    Returns:
        A list containing the most recent portion of the chat history,
        limited to a maximum of 2048 tokens.
    """

    # Initialize a variable to keep track of the total number of tokens
    sum_tokens = 0

    # Reverse the history list to process from the most recent conversation
    history.reverse()

    # Iterate through each QnA pair in the reversed history
    for i, qna in enumerate(history):
        # Count the number of tokens in the current QnA pair (question and answer comb
        sum_tokens += len(str(qna).split())

        # Check if the total token count exceeds the limit
        if sum_tokens > 2048:
            # Reverse the history list back to its original order
            history.reverse()
            # Return the portion of the history exceeding the limit (most recent)
            return history[i:]

    # If the total token count doesn't exceed the limit, reverse the history back and
    history.reverse()
    return history
```

In [ ]:
```python
def chat_(chat_history, question):
    """
    This function simulates a conversation with a chatbot portraying a Mental Health P

    Args:
        chat_history: A list of dictionaries, where each dictionary represents
            a question-answer pair from the previous chatbot interactions.
        question: The new question to be posed to the chatbot.
```

```
    Yields:
        A list containing the updated chat history with the latest question and respon
    """

    global chat_history_

    # Check if there is existing chat history
    if chat_history_:
        # Construct the prompt template incorporating the previous chat history
        template = (
            "You are a Mental Health Professional who helps people with mental disorde
            "Based on this chat history we've been discussing:\n"
            + str(cut_history(chat_history_)).replace("{", "<").replace("}", ">")
            + "\n Answer this question: \"{ques1}\" with a conversational answer."
        )

        # Define the prompt template with an input variable "ques1" for the question
        prompt = PromptTemplate(input_variables=["ques1"], template=template)

        # Create an LLM chain using the pre-defined LLM object and the prompt
        chain = LLMChain(llm=llm, prompt=prompt)

    else:
        # If there is no previous chat history, use a simpler template
        template = (
            "You are a Mental Health Professional who helps people with mental disorde
            + "\n Answer this question: \"{ques1}\" with a conversational answer."
        )

        # Define the prompt template with an input variable "ques1" for the question
        prompt = PromptTemplate(input_variables=["ques1"], template=template)

        # Create an LLM chain using the pre-defined LLM object and the prompt
        chain = LLMChain(llm=llm, prompt=prompt)

    # Print the constructed prompt template for potential debugging or inspection
    print(prompt.template)

    # Run the LLM chain with the provided question and get the response
    bot_response = chain.run(question)

    # Initialize an empty string to store the final response
    response = ""

    # Append the latest question and answer to the chat history
    chat_history_.append({"Question": question, "Answer": bot_response})

    # Combine the individual characters of the response into a string
    for letter in ''.join(bot_response):
        response += letter + ""

    # Yield the updated chat history with the latest question and response
    yield chat_history + [(question, response)]
```

In [ ]:

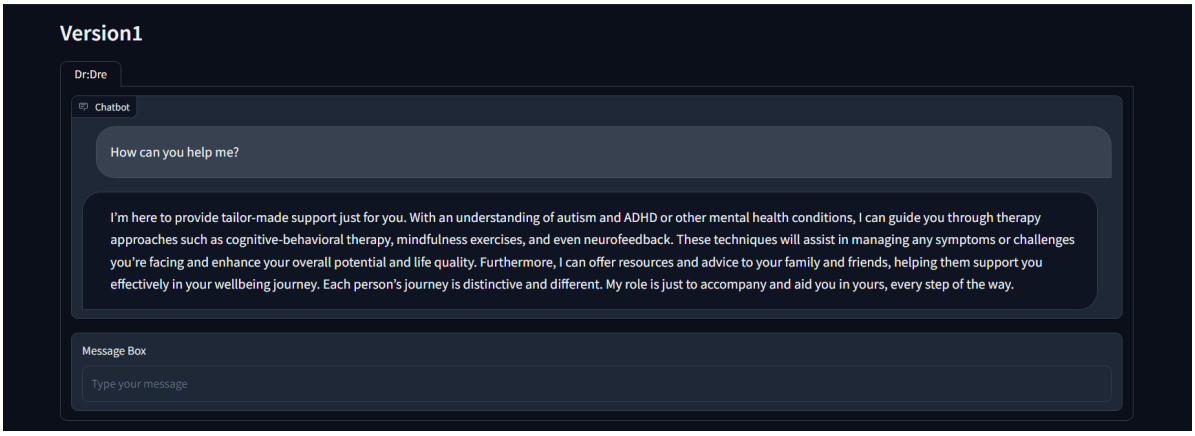
```
# Create a Gradio Blocks context manager
with gr.Blocks() as demo:

    # Define a Markdown title for the first block
    gr.Markdown('# Version1')
```

```
    # Create a tab named "Dr:Dre"
    with gr.Tab("Dr:Dre"):

        # Initialize a Gradio Chatbot component
        chatbot = gr.Chatbot()

        # Create a Textbox component with placeholder text and label
        message = gr.Textbox(placeholder='Type your message', label='Message Box')

        # Define a submit function for the Textbox
        # This function takes three arguments:
        #   - `chat_`: The chat history function
        #   - `[chatbot, message]`: A list containing the chatbot and message componen
        #   - `chatbot`: The chatbot component itself (for potential internal actions)
        message.submit(chat_, [chatbot, message], chatbot)

# Queue the Gradio app and launch it
demo.queue().launch(debug=True, share=True, server_port=8000)
```

```
Running on local URL:  http://127.0.0.1:8000
Running on public URL: https://f4ed5a43cb8584958d.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades (NE
W!), check out Spaces: https://huggingface.co/spaces
```

```
You are a Mental Health Professional who helps people with mental disorders like auti
sm and ADHD.
 Answer this question: "{ques1}" with a conversational answer.
You are a Mental Health Professional who helps people with mental disorders like auti
sm and ADHD.
Based on this chat history we've been discussing:
[<'Question': 'How can you help me?', 'Answer': "As a Mental Health Professional, I s
pecialize in providing support to individuals with conditions like autism and ADHD. O
nce we understand your specific needs, I can utilize a wide range of approaches, such
as cognitive-behavioral therapy, psychoeducational counseling, mindfulness strategie
s, neurofeedback, and more, which are personalized for each individual. The aim here
is to help you manage the symptoms and challenges you're facing, maximize your potent
ial, and improve your quality of life. Additionally, I can also provide support and r
esources for your loved ones, to ensure they understand and can assist in your journe
y towards wellbeing. Remember, each person's path is unique, and I'm here to walk on
that journey with you and help wherever I can.">]
 Answer this question: "{ques1}" with a conversational answer.
Keyboard interruption in main thread... closing server.
Killing tunnel 127.0.0.1:8000 <> https://f4ed5a43cb8584958d.gradio.live
```

Out[ ]:



Version two demo:

Here, we will be exploring how we can benefit from VectorDBs to make our pre-trained LLMs more instructed and specified for our specific need and custom data.

Vector Databases (VectorDBs) and their use in Pre-trained LLMs

1. What are VectorDBs?

Vector databases (VectorDBs) are a specialized type of database designed to store and efficiently search high-dimensional vectors. These vectors represent information like word embeddings, document embeddings, or image features, where each dimension captures a specific aspect of the data. VectorDBs offer fast retrieval of similar items based on their vector representations.

2. How can VectorDBs be used with Pre-trained LLMs?

Pre-trained LLMs, like GPT-3.5, are powerful language models trained on massive amounts of text data. However, they can benefit from additional information sources during tasks like question answering or generating different creative text formats. Here's how VectorDBs can be used with pre-trained LLMs:

a) Retrieval Augmentation (RAG):

- A VectorDB can store pre-computed vector representations of documents or knowledge bases relevant to the LLM's domain.
- When the LLM receives a user query, the VectorDB can be used to retrieve the most relevant documents or information based on the query's vector representation.
- The retrieved information can then be provided to the LLM as context, potentially improving the focus and accuracy of its response by grounding it in relevant factual knowledge.

b) Information Augmentation:

- Similar to RAG, VectorDBs can store pre-computed factual information or knowledge graphs represented as vectors.
- During LLM reasoning or generation tasks, the LLM can access and integrate this information from the VectorDB to enhance the coherence and factual correctness of its output.

c) Personalization:

- VectorDBs can be used to store user profiles or preferences represented as vectors.
- When a user interacts with the LLM, their profile vector can be used to personalize the LLM's responses or tailor its communication style to their interests.

3. Benefits of using VectorDBs with Pre-trained LLMs:

- **Improved Accuracy:** By providing access to relevant factual information, VectorDBs can help LLMs generate more accurate and factually grounded responses.
- **Enhanced Focus:** Retrieved information from the VectorDB can guide the LLM's reasoning and generation process, leading to more focused and relevant outputs.
- **Personalization:** User profiles stored in VectorDBs can personalize the LLM's interaction with each user.

4. Challenges and Considerations:

- **Building and Maintaining the VectorDB:** Creating and maintaining a high-quality VectorDB requires domain expertise and ongoing effort to ensure its content is accurate and up-to-date.
- **Integration with LLMs:** Integrating VectorDBs with LLMs requires careful design and consideration of how retrieved information is best presented and utilized by the LLM.

Overall, VectorDBs offer a promising approach for enhancing the capabilities of pre-trained LLMs by providing them with structured and efficient access to relevant information, so let's see how this approach will go by starting with preprocessing our data sources to fit the VectorDB we're going to build.

PDF Information Extraction

The following Python script extracts information from our PDF source files and stores it in a structured format.

Functionality:

- The script utilizes the `PyPDF2` library to read and process PDF content.
- It iterates through a predefined list of PDF file paths.
- For each PDF file:
  - It opens the file in binary read mode to ensure proper handling of potentially non-ASCII characters.
  - It creates a `PdfReader` object to access the PDF's pages and content.
  - It loops through each page in the PDF.
  - For each page:
    - It extracts the text content using the `extract_text` method.
    - It splits the page text into sentences based on the pattern ". \n" (full stop followed by a new line). This assumes a specific sentence ending format, which might need adjustment depending on the PDFs.
    - It loops through each extracted sentence.
    - It filters out sentences with less than 10 words, potentially excluding incomplete or irrelevant information.
    - For each valid sentence (10+ words):
      - It creates a dictionary to store information about the sentence.
        - The dictionary includes:
          - `information`: The actual sentence content itself.
          - `source_file_name`: The name of the PDF file the sentence was extracted from (using `os.path.basename`).
          - `page_number`: The page number where the sentence was found (starts from 1).
      - It appends the information dictionary to a global list named `pdf_pages`.

Output:

- The script builds a list named `pdf_pages` containing dictionaries for each valid sentence found in the processed PDFs.

In [ ]:
```
from PyPDF2 import PdfReader
import os

# Define a list to store information extracted from each page
```

```
pdf_pages = []

# Specify the paths to the PDF files to be processed
pdf_file_paths = [
    r"D:\Graduation project\Diagnostic and statistical manual of mental disorders _ DS
    r"D:\Graduation project\19-mh-8084-autismspectrumdisorder.pdf"
]

# Loop through each PDF file path
for pdf_file_path in pdf_file_paths:
    # Open the PDF file in binary read mode
    with open(pdf_file_path, 'rb') as pdf_file:
        # Create a PdfReader object to access the PDF content
        reader = PdfReader(pdf_file)

        # Loop through each page in the PDF
        for page_number in range(len(reader.pages)):
            # Extract the text content of the current page
            page_text = reader.pages[page_number].extract_text()

            # Split the page text into sentences based on full stops followed by a new
            sentences = page_text.split('.\n')

            # Loop through each sentence
            for content in sentences:
                # Filter out sentences with less than 10 words (potentially incomplete
                if len(content.split()) >= 10:
                    # Create a dictionary to store information about the current sente
                    pdf_page_info = {
                        'information': content,  # The sentence itself
                        'source_file_name': os.path.basename(pdf_file_path),  # Name o
                        'page_number': page_number + 1  # Page number (starts from 1)
                    }

                    # Append the dictionary to the list of page information
                    pdf_pages.append(pdf_page_info)
```

In [ ]:
```
pdf_pages[1000] # Displaying a sample information.
```

```
{'information': 'If an individual with cyclothymic disorder su bsequently (i.e., afte
r the initial 2 years in\nadults or 1 year in children or adolescents) experiences a
major depressive, manic, or hy-\npomanic episode, the diagnosis changes to majo r dep
ressive disorder , bipolar I disorder,\nor other specified or unspecified bipolar and
related disorder (subclassified as hypomanic\nepisode without prior major depressive
episod e), respectively, and the cyclothymic disor-\nder diagnosis is dropped',
 'source_file_name': 'Diagnostic and statistical manual of mental disorders _ DSM-5 (
PDFDrive.com ).pdf',
 'page_number': 185}
```

In [ ]:
```
len(pdf_pages) # Number of pages.
```

```
5867
```

In [ ]:
```
import pandas as pd

df = pd.DataFrame(pdf_pages) # Converting the pdf information list to a pandas datafra
df.to_csv('preprocessed_dataset.csv',index=False) # Saving the preprocessed PDF files
df
```

	information	source_file_name	page_number
0		Diagnostic and statistical manual of mental di...	1
1	DIAGNOSTIC AND STATISTICAL \nMANUAL OF\nMENTAL...	Diagnostic and statistical manual of mental di...	2
2	American Psychiatric Association\nOfficers 201...	Diagnostic and statistical manual of mental di...	3
3	PRESIDENT -ELECT JEFFREY A. L IEBERMAN , M.D	Diagnostic and statistical manual of mental di...	3
4	TREASURER DAVID FASSLER , M.D	Diagnostic and statistical manual of mental di...	3
...	...	...	...
7951	◅NIMH requests that non-federal organizations...	19-mh-8084-autismspectrumdisorder.pdf	7
7952	◅The addition of non-federal government logos...	19-mh-8084-autismspectrumdisorder.pdf	7
7953	◅Images pictured in NIMH publications are of ...	19-mh-8084-autismspectrumdisorder.pdf	7
7954	If you have questions regarding these guidelin...	19-mh-8084-autismspectrumdisorder.pdf	7
7955	For More Information\nNIMH website \nwww.nimh....	19-mh-8084-autismspectrumdisorder.pdf	8

7956 rows × 3 columns

In [ ]:
```
df.iloc[1000].information
```

```
'Differential Diagnosis\nOther mental disorders and medical conditions. A wide variet
y of psychiatric and med-\nical conditions can manifest with psychotic and mood sympt
oms that must be considered\nin the differential diagnosis of  schizoaffective disord
er. Th ese include psychotic disorder\ndue to another medical condition; delirium; ma
jor neurocognitive disorder; substance/\nmedication-induced psychotic disorder or neu
rocognitive disorder; bipolar disorders\nwith psychotic features; major depressive di
so rder with psychotic features; depressive or\nbipolar disorders with catatonic feat
ures; schi zotypal, schizoid, or paranoid personality\ndisorder; brief psychotic diso
rder; schizophre niform disorder; schizophrenia; delusional\ndisorder; and other spec
ified and unspecified schizophrenia spectrum and other psychotic\ndisorders. Medical
conditions and substance use can present with a combination of psy-\nchotic and mood
symptoms, and thus psychoti c disorder due to another medical condition\nneeds to be
excluded. Distinguishing schizoaffective disorder from schizophrenia and\nfrom depres
sive and bipolar diso rders with psychotic features  is often difficult. Criterion\nC
is designed to separate schizoaffective di sorder from schizophrenia, and Criterion B
is\ndesigned to distinguish schizoaffective diso rder from a depressive or bipolar di
sorder\nwith psychotic features. More specifically, schizoaffective disorder can be d
istinguished\nfrom a depressive or bipolar disorder with psyc hotic features due to t
he presence of prom-\ninent delusions and/or hallucinations for at least 2 weeks in t
he absence of a major mood\nepisode. In contrast, in depressive or bipo lar disorders
with psychotic features, the psy-\nchotic features primarily occur during the mo od e
pisode(s). Because the relative propor-\ntion of mood to psychotic symptoms may ch an
ge over time, the appropriate diagnosis\nmay change from and to schizoaffective disor
d er (e.g., a diagnosis of schizoaffective dis-\norder for a severe and prominent maj
or depre ssive episode lasting 3 months during the\nfirst 6 months of a persistent ps
ychotic illnes s would be changed to schizophrenia if active\npsychotic or prominent
residual symptoms pe rsist over several years without a recurrence\nof another mood e
pisode)'
```

In [ ]:
```
df.iloc[1001].information
```

Out[ ]:
```
'The cyclothymic disorder diagnosis is not made  if the pattern of mood swings is bet
ter\nexplained by schizoaffective disorder, schizophrenia, schizophreniform disorder,
delu-\nsional disorder, or other specified and un specified schizophrenia spectrum an
d other\npsychotic disorders (Criterion D), in which c ase the mood symptoms are cons
idered asso-\nciated features of the psychoti c disorder. The mood disturbanc e must
also not be attribut-\nable to the physiological effe cts of a substance (e.g., a dr
ug of abuse, a medication) or\nanother medical condition (e.g., hyperthyroidis m) (Cr
iterion E). Although some individ-\nuals may function particularly  well during some
of the pe riods of hypomania, over the\nprolonged course of the disorder , there must
be clinically significant distress or impair-\nment in social, occupational, or other
import ant areas of functioning as a result of the\nmood disturbance (Criterion F). T
he impairment may develop as a result of prolonged pe-\nriods of cyclical, often unpr
edictable mood ch anges (e.g., the individual may be regarded\nas temperamental, mood
y, unpredicta ble, inconsistent , or unreliable)'
```

In [ ]:
```
import pickle as pk

with open('pages_lst.pkl','wb') as f:
    pk.dump(pdf_pages,f) # Saving the preprocessed PDF files as a list object.
```

Now after we've extracted and preprocessed our data sources, let's get the embeddings of our data... wait, what are embeddings?

Embedding Models, VectorDBs, and all-MiniLM-L6-v2

Here's a breakdown of embedding models, their use in building VectorDBs, and the all-MiniLM-L6-v2 model:

1. Embedding Models:

Embedding models are a type of neural network architecture used to transform data (text, images, etc.) into numerical representations called embeddings. These embeddings capture the semantic meaning and relationships within the data in a high-dimensional space. Here are some key points about embedding models:

- **Input and Output:** They take raw data (e.g., a sentence) as input and generate a vector of numbers (embedding) as output.
- **Dimensionality:** The number of dimensions in the embedding vector determines the level of detail it can capture. Higher dimensions allow for more nuanced representations.
- **Similarity:** Similar data points will have embeddings closer together in the vector space, while dissimilar data points will be further apart.
- **Pre-trained vs. Fine-tuned:** Embedding models can be pre-trained on massive datasets or fine-tuned on specific datasets for improved performance in a particular domain.

2. Building VectorDBs with Embedding Models:

VectorDBs rely on embedding models to create efficient representations of data for faster search and retrieval. Here's how they work together:

- **Data Processing:** Text data is cleaned and preprocessed before feeding it into the embedding model.
- **Embedding Generation:** The embedding model takes each piece of data (e.g., sentence, document) and generates its corresponding vector representation.
- **Storage in VectorDB:** These vector representations are then stored in the VectorDB alongside the original data or metadata for easy retrieval.
- **Retrieval:** When a query is submitted, the VectorDB can compare its vector representation with the stored embeddings. Documents or information with embeddings closest to the query vector are retrieved as most relevant.

3. The all-MiniLM-L6-v2 Model:

The all-MiniLM-L6-v2 model is a pre-trained sentence embedding model specifically designed for generating high-quality sentence embeddings. Here's what makes it stand out:

- **Underlying Architecture:** It's based on the MiniLM model, a smaller and faster variant of the well-known BERT model.
- **Pre-training Data:** It's trained on a massive dataset of sentence pairs, learning to identify semantic similarity between sentences.
- **Focus on Sentences:** Unlike some embedding models that work on words, all-MiniLM-L6-v2 focuses on capturing the meaning of entire sentences.
- **Community Developed:** This model is a result of community efforts and collaboration, making it readily available and fostering further development.

Overall:

Embedding models like all-MiniLM-L6-v2 play a crucial role in building VectorDBs. They efficiently convert textual data into a numerical format that allows for fast and accurate similarity searches. This empowers VectorDBs to be a valuable source of information for tasks like retrieval augmentation and information augmentation in pre-trained LLMs.

Embedding Models, Vector Databases, and all-MiniLM-L6-v2: A Deep Dive with Resources

Now that we've tackled data extraction and preprocessing, let's delve into the fascinating world of embeddings! But before we jump in, what exactly are embeddings?

Embedding Models: Capturing Meaning in Numbers

Embedding models are a powerful type of neural network architecture that transform data like text, images, and even audio into numerical representations called embeddings. These embeddings act like condensed summaries, capturing the essence and relationships within the data in a high-dimensional space. Imagine them as unique fingerprints for your data points!

Here are some key concepts to grasp embedding models:

- **Input and Output:** They take raw, unprocessed data (think a sentence or an image) as input and generate a vector of numbers (the embedding) as output.
- **Dimensionality:** The number of dimensions in this embedding vector determines the level of detail it can capture. Higher dimensions allow for more nuanced representations, but also require more computational resources.
- **Similarity:** The magic lies in how embeddings represent relationships. Similar data points, like sentences with similar meanings, will have embeddings closer together in the vector space. Conversely, dissimilar data points will be further apart. This allows for efficient similarity searches – finding data points with characteristics close to a given query.
- **Pre-trained vs. Fine-tuned:** Embedding models can be:
  - **Pre-trained:** Trained on massive, general-purpose datasets, making them versatile for various tasks.
  - **Fine-tuned:** Further trained on a specific dataset to improve performance in a particular domain (e.g., legal documents or medical research papers).

For a deeper understanding of embedding models, here are some resources:

- **The Illustrated Transformer:** This interactive website explains the Transformer architecture, a core component used in many modern embedding models
- **A Primer on Neural Word Embeddings:** https://rcpedia.stanford.edu/topicGuides/textProcessingWord_Embeddings.html (This Stanford University resource provides a comprehensive overview of word embedding models, a specific type of embedding model used for text data)

2. Building VectorDBs with Embedding Models: The Power of Embeddings

VectorDBs rely on embedding models to create efficient representations of data for faster search and retrieval. Here's how they work together:

- **Data Processing:** Text data is cleaned and preprocessed before feeding it into the embedding model.
- **Embedding Generation:** The embedding model takes each piece of data (e.g., sentence, document) and generates its corresponding vector representation.
- **Storage in VectorDB:** These vector representations are then stored in the VectorDB alongside the original data or metadata for easy retrieval.
- **Retrieval:** When a query is submitted, the VectorDB can compare its vector representation with the stored embeddings. Documents or information with embeddings closest to the query vector are retrieved as most relevant.

3. The all-MiniLM-L6-v2 Model: A Pre-Trained Champion for Sentence Embeddings

The all-MiniLM-L6-v2 model is a pre-trained sentence embedding model specifically designed for generating high-quality sentence embeddings. Here's what makes it stand out:

- **Underlying Architecture:** It's based on the MiniLM model, a smaller and faster variant of the well-known BERT model.
- **Pre-training Data:** It's trained on a massive dataset of sentence pairs, learning to identify semantic similarity between sentences.
- **Focus on Sentences:** Unlike some embedding models that work on words, all-MiniLM-L6-v2 focuses on capturing the meaning of entire sentences.
- **Community Developed:** This model is a result of community efforts and collaboration, making it readily available and fostering further development.

You can read more about all-MiniLM-L6-v2 Model and BERT model from the following articles:

- https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
- https://towardsdatascience.com/keeping-up-with-the-berts-5b7beb92766

In [ ]:
```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer('all-MiniLM-L6-v2') # Loading our embedding model.

# Sentences we'd like to encode
sentences = ['American Psychiatric Association']

# Sentences are encoded by calling model.encode()
embeddings = model.encode(sentences).tolist()
embeddings[0][:10] # displaying a chunk of the encoding/embedded sentence
```

Out[ ]:
```
[-0.022938288748264313,
 -0.0636344626545906,
 -0.08395121991634369,
 0.04752057045698166,
 -0.0468604639172554,
 0.11838657408952713,
 0.04775754362344742,
 -0.02057965099811554,
 -0.005866041872650385,
 -0.02532806620001793]
```

Now that we've got our embedding model loaded and ready to encode our sentences, let's connect to our VectorDB that we will store all embedded sentences in, we'll be doing so by importing the Pinecone library and using its functionalities and classes to get our VectorDB up and ready to be used, we've talked about Pinecone and VectorDBs in previos sections on the book, feel free to go back and read them again if you need to before diving into the documentation and code below.

Setting Up our Pinecone Vector Database

Before we delve further into the code, let's set up our Pinecone Vector Database. Pinecone offers a cloud-based service, so you don't need to manage infrastructure ourself. Here's a step-by-step guide:

1. **Create a Pinecone Account:**
    - Head over to the Pinecone website: https://www.pinecone.io/
    - Click on "Sign Up" and follow the instructions to create a free account.
2. **Create a Project:**
    - Once logged in, navigate to the "Projects" section.
    - Click on "Create Project" and provide a descriptive name for our project (e.g., "My Text Retrieval Project").
3. **Obtain our API Key:**
    - After creating our project, locate the "API Keys" section within the project settings.
    - Copy the provided API key. You'll need this key to interact with Pinecone's API from our code.

Important Note:

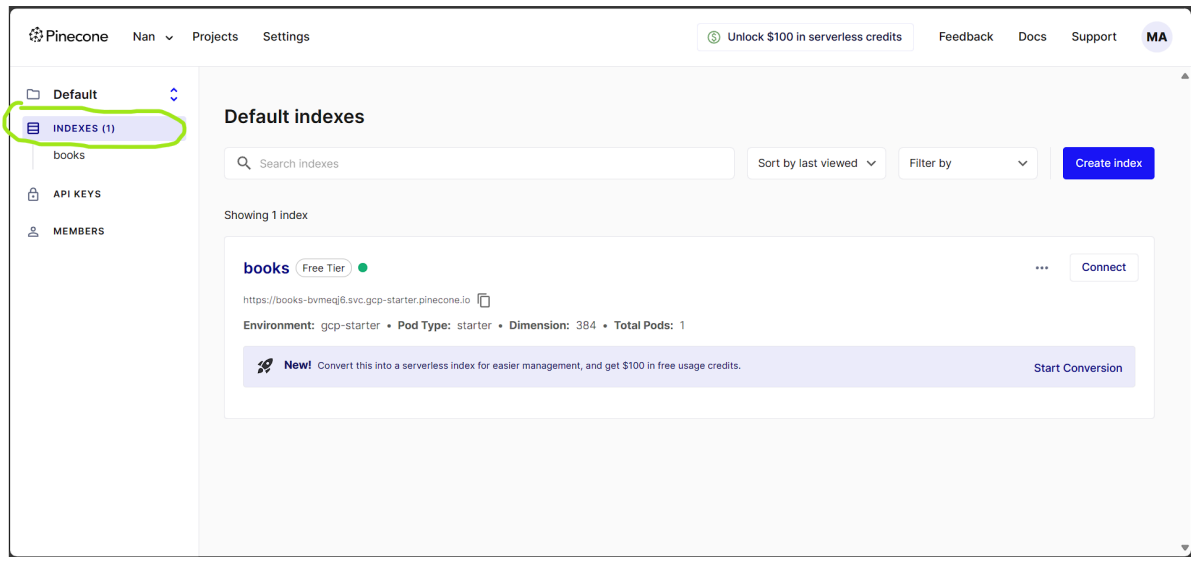
- Securely store our API key. Avoid sharing it publicly.

In [ ]:
```python
from pinecone import Pinecone

# initialize connection to pinecone (get API key at app.pinecone.io)
api_key = ''

# configure client
pc = Pinecone(api_key=api_key)
```

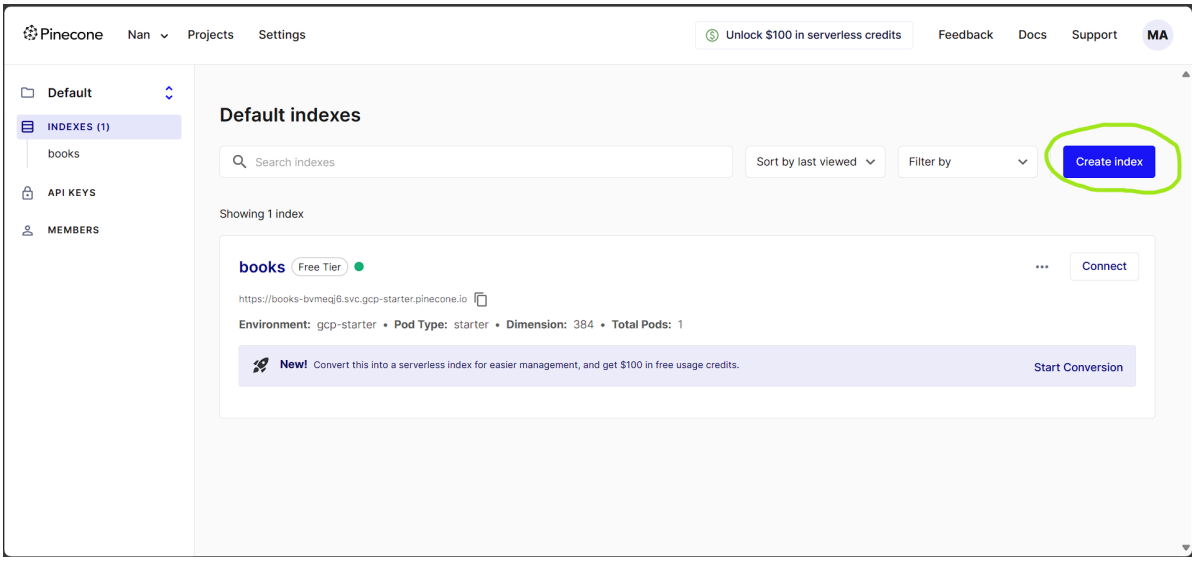
In [ ]:
```python
# Import the ServerlessSpec class from the pinecone library
from pinecone import ServerlessSpec

# Define a ServerlessSpec object named 'spec'
spec = ServerlessSpec(

    # Specify the cloud provider as "aws" (Amazon Web Services)
    cloud="aws",

    # Specify the desired region within AWS as "us-west-2" (US West 2)
    region="us-west-2"
)
```

Now follow the instructions bellow to create your VectorDB:

- Got to the `Indexes` tap:

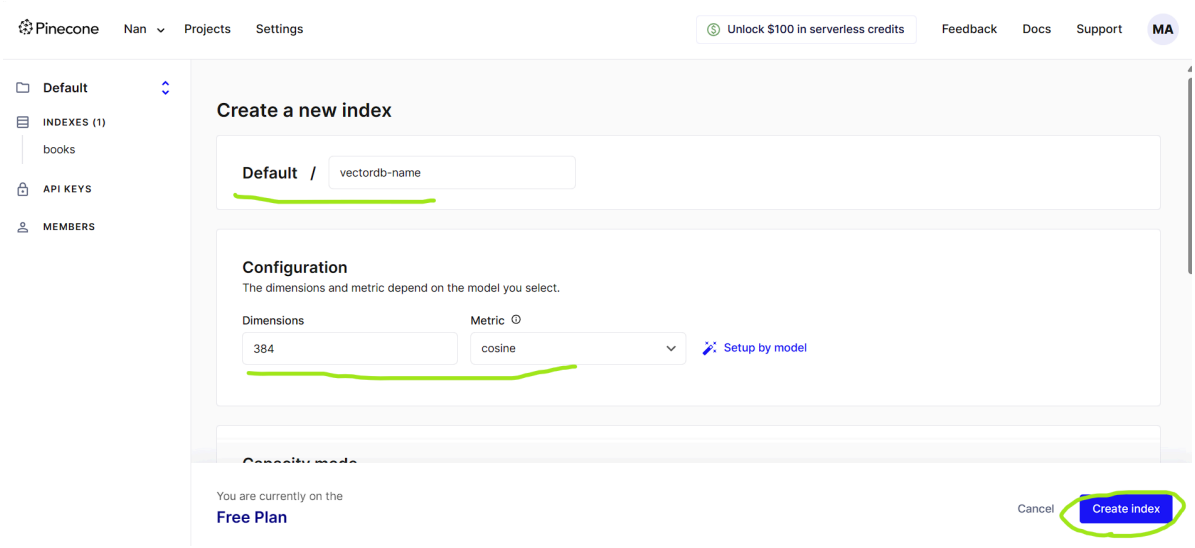


- Click `Create index`:

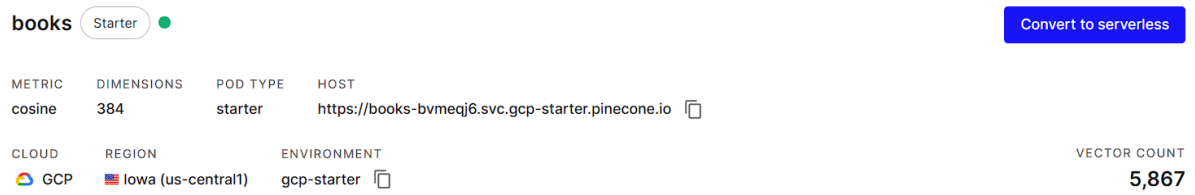


- Write the name of you VectorDB, the number of your embedding model demintions (in our case it's 384), and the metric (this metric measures how well the model captures the

semantic relationships between sentences.)



Now after you know how to create your own VectorDB, here's the VectorDB we've created for this project:



Let's connect to it via the code:

In [ ]:
```
import time

index_name = 'books'

# connect to index
index = pc.Index(index_name)
time.sleep(1)
# view index stats
index.describe_index_stats() # At first, it should return empty values for the vector_
```

Out[ ]:
```
{'dimension': 384,
 'index_fullness': 0.05867,
 'namespaces': {'': {'vector_count': 5867}},
 'total_vector_count': 5867}
```

Let's assume that our VectorDB is still empty and we need to upload our embedded data to it, here's the code for that:

In [ ]:
```
# Import the tqdm library for progress bars
from tqdm.auto import tqdm

# Create a copy of the dataframe for efficient iteration
data = df.copy()  # Avoid modifying the original dataframe

# Set the batch size for processing data in chunks
```

```
batch_size = 100

# Iterate through the data in batches
for i in tqdm(range(0, len(data), batch_size)):  # Display a progress bar

    # Calculate the end index for the current batch
    i_end = min(len(data), i + batch_size)

    # Get the current batch of data from the dataframe
    batch = data.iloc[i:i_end]

    # Generate unique IDs using file name, page number, and index within the batch
    ids = [f"{x['source_file_name']}-{x['page_number']}-{i}" for i, x in batch.iterrow

    # Extract the text to be embedded from the batch
    texts = [x['information'] for _, x in batch.iterrows()]

    # Generate embeddings for the text
    embeds = model.encode(texts).tolist()

    # Prepare metadata to be stored in Pinecone alongside the embeddings
    metadata = [
        {'information': x['information'], 'page_number': x['page_number'], 'source_fil
        for i, x in batch.iterrows()
    ]

    # Add the embeddings and metadata to Pinecone in a batch
    index.upsert(vectors=zip(ids, embeds, metadata))
```

Now let's build some helper functions to cut and augment our prompts' context since the maximum input tokens size for most OpenAI's LLMs is 4096 token per requist/message, and we don't wish to exceed that limit to avoid any errors.

In [ ]:

```
def cut_context(context):
    """
    This function cuts off a conversation history if it exceeds a maximum length
    in terms of the total number of tokens (words).

    Args:
        context: A list of QnA pairs, where each QnA pair represents
            a question and answer exchange in the previous chatbot interactions.

    Returns:
        A list containing the most recent portion of the conversation history,
        limited to a maximum of 2048 tokens.
    """

    sum_tokens = 0
    context.reverse()  # Reverse the list for processing most recent conversation firs

    # Iterate through each QnA pair in the reversed history
    for i, qna in enumerate(context):
        sum_tokens += len(str(qna).split())  # Count tokens in the current QnA pair

        # Check if the total token count exceeds the limit
        if sum_tokens > 2048:
            context.reverse()  # Reverse the list back to its original order
            return context[i:]  # Return the portion exceeding the limit (most recent)
```

```
    context.reverse()  # Reverse the list back if no truncation occurred
    return context  # Return the entire context if it's within the limit


def augment_prompt(query: str, history: list):
    """
    This function creates an augmented prompt for the chatbot by incorporating
    knowledge base search results and chat history context.

    Args:
        query: The user's query to be posed to the chatbot.
        history: A list of dictionaries, where each dictionary represents
            a question-answer pair from the previous chatbot interactions.

    Returns:
        A string containing the augmented prompt for the chatbot.
    """

    # Retrieve top 4 results from the VectorDB/knowledge base based on encoded query a
    results = cut_context([match['metadata'] for match in index.query(4, model.encode(

    # Combine the retrieved knowledge base information into a single string
    source_knowledge = "\n".join([str(x) for x in results])

    # Construct the augmented prompt by incorporating contexts and the query
    augmented_prompt = f"""You are a Mental Health Professional who helps people with
Using the contexts and chat history below, answer the query.

Contexts:
{source_knowledge}

Chat history:
{str(history)}

Query: {query}"""

    return augmented_prompt
```

In [ ]:
```
ap = augment_prompt('Have I ADHD?',[])

print(ap, len(ap.split()))
```

```
You are a Mental Health Professional who helps people with mental disorders like auti
sm and ADHD.
Using the contexts and chat history below, answer the query.

    Contexts:
    {'information': 'There may be an elevated likelihood of  obesity among individual
s with ADHD', 'page_number': 108.0, 'source_file_name': 'Diagnostic and statistical m
anual of mental disorders _ DSM-5 ( PDFDrive.com ).pdf'}
{'information': 'ADHD is a neurodevelopmental disorder defined by impairing levels of
inattention, dis-\norganization, and/or hyperactivity-impulsivity. Inattention and di
sorganization entail inabil-\nity to stay on task, seeming not to listen, and lo sing
materials, at leve ls that are inconsistent\nwith age or developmental level.  Hypera
ctivity-impulsivity entails  overactivity, fidgeting, in-\nability to stay seated, in
truding into other people's activities , and inability to wait—symptoms\nthat are exc
essive for age or developmental level. In childhood, ADHD frequently overlaps\nwith d
isorders that are often considered to be "externalizing disorders," such as oppositio
nal\ndefiant disorder and conduct di sorder. ADHD often persists in to adulthood, wit
h resultant\nimpairments of social, academic and occupational functioning', 'page_num
ber': 77.0, 'source_file_name': 'Diagnostic and statistical manual of mental disorder
s _ DSM-5 ( PDFDrive.com ).pdf'}
{'information': 'Medication-induced symptoms of ADHD. Symptoms of inattention, hypera
ctivity, or\nimpulsivity attributable to the use of medication (e.g., bron chodilator
s, isoniazid, neuro-\nleptics [resulting in akathisia], thyroid replacement medicatio
n) are diagnosed as other\nspecified or unspecified other (or un known) substance–rel
ated disorders', 'page_number': 110.0, 'source_file_name': 'Diagnostic and statistica
l manual of mental disorders _ DSM-5 ( PDFDrive.com ).pdf'}
{'information': 'Visual and hearing impairments, metabolic abnormalities, sleep  diso
rders, nutritional de-\nficiencies, and epilepsy should be considered as possible inf
luences on ADHD symptoms', 'page_number': 107.0, 'source_file_name': 'Diagnostic and
statistical manual of mental disorders _ DSM-5 ( PDFDrive.com ).pdf'}

    Chat history:
    []

    Query: Have I ADHD? 271
```

Looks like a great prompt for our question and LLM. Let's move on to the LLM code, we won't explain a lot from now on since everything was already documneted in Version one documentation.

In [ ]:

```python
import os

os.environ["OPENAI_API_KEY"] = ""  # Replace with your actual OpenAI API key

from langchain.llms import OpenAI

# Initialize OpenAI client for interacting with GPT-3.5-turbo-instruct model
llm = OpenAI(model_name="gpt-3.5-turbo-instruct")

chat_history_ = []


def cut_history(history):
    """
    This function cuts off a conversation history if it exceeds a maximum length
    in terms of the total number of tokens (words).

    Args:
        history: A list of QnA pairs, where each QnA pair represents
```

```
            a question and answer exchange in the previous chatbot interactions.

    Returns:
        A list containing the most recent portion of the conversation history,
        limited to a maximum of 2048 tokens.
    """

    sum_tokens = 0
    history.reverse()  # Reverse the list for processing most recent conversation firs

    # Iterate through each QnA pair in the reversed history
    for i, qna in enumerate(history):
        sum_tokens += len(str(qna).split())  # Count tokens in the current QnA pair

        # Check if the total token count exceeds the limit
        if sum_tokens > 2048:
            history.reverse()  # Reverse the list back to its original order
            return history[i:]  # Return the portion exceeding the limit (most recent)

    context.reverse()  # This line was accidentally included from a previous version,
    return history  # Return the entire context if it's within the limit
```

In [ ]:

```
# Define the user's question
question = 'Have I ADHD?'

# Prepare the chat history
#   - Slice the last 9 entries from chat_history_ (assuming it's a list)
#   - Call cut_history to potentially limit the conversation history length
#   - Convert the potentially truncated history to a string
#   - Escape curly braces to avoid issues with prompt formatting
prompt = augment_prompt(question, str(cut_history(chat_history_[-9:])).replace('{', '<

# Print the constructed prompt for debugging or logging purposes
print(prompt)

# Send the prompt to the OpenAI API using the llm object
bot_response = llm(prompt)

# Update the chat history with the user's question and the bot's response
chat_history_.append({"Question": question, "Answer": bot_response})
```

```
You are a Mental Health Professional who helps people with mental disorders like auti
sm and ADHD.
Using the contexts and chat history below, answer the query.

    Contexts:
    {'information': 'There may be an elevated likelihood of  obesity among individual
s with ADHD', 'page_number': 108.0, 'source_file_name': 'Diagnostic and statistical m
anual of mental disorders _ DSM-5 ( PDFDrive.com ).pdf'}
{'information': 'ADHD is a neurodevelopmental disorder defined by impairing levels of
inattention, dis-\norganization, and/or hyperactivity-impulsivity. Inattention and di
sorganization entail inabil-\nity to stay on task, seeming not to listen, and lo sing
materials, at leve ls that are inconsistent\nwith age or developmental level.  Hypera
ctivity-impulsivity entails  overactivity, fidgeting, in-\nability to stay seated, in
truding into other people's activities , and inability to wait—symptoms\nthat are exc
essive for age or developmental level. In childhood, ADHD frequently overlaps\nwith d
isorders that are often considered to be "externalizing disorders," such as oppositio
nal\ndefiant disorder and conduct di sorder. ADHD often persists in to adulthood, wit
h resultant\nimpairments of social, academic and occupational functioning', 'page_num
ber': 77.0, 'source_file_name': 'Diagnostic and statistical manual of mental disorder
s _ DSM-5 ( PDFDrive.com ).pdf'}
{'information': 'Medication-induced symptoms of ADHD. Symptoms of inattention, hypera
ctivity, or\nimpulsivity attributable to the use of medication (e.g., bron chodilator
s, isoniazid, neuro-\nleptics [resulting in akathisia], thyroid replacement medicatio
n) are diagnosed as other\nspecified or unspecified other (or un known) substance–rel
ated disorders', 'page_number': 110.0, 'source_file_name': 'Diagnostic and statistica
l manual of mental disorders _ DSM-5 ( PDFDrive.com ).pdf'}
{'information': 'Visual and hearing impairments, metabolic abnormalities, sleep  diso
rders, nutritional de-\nficiencies, and epilepsy should be considered as possible inf
luences on ADHD symptoms', 'page_number': 107.0, 'source_file_name': 'Diagnostic and
statistical manual of mental disorders _ DSM-5 ( PDFDrive.com ).pdf'}

    Chat history:
    []

    Query: Have I ADHD?
c:\Users\LAPTOP WORLD\AppData\Local\Programs\Python\Python39\lib\site-packages\langch
ain_core\_api\deprecation.py:117: LangChainDeprecationWarning: The function `__call__
` was deprecated in LangChain 0.1.7 and will be removed in 0.2.0. Use invoke instead.
  warn_deprecated(
```

In [ ]:
```
bot_response
```

```
'\n\nBased on the information provided, I cannot determine if you have ADHD. ADHD is
a neurodevelopmental disorder characterized by symptoms such as inattention, hyperact
ivity, and impulsivity. It is important to consult with a mental health professional
for a proper evaluation and diagnosis. Other factors, such as visual and hearing impa
irments, metabolic abnormalities, sleep disorders, and nutritional deficiencies, shou
ld also be considered as possible influences on ADHD symptoms.'
```

A great response, I'd say. Let's see how the deployment of this version will go!

Version two Deployment:

In [ ]:
```
# Import libraries for interacting with Pinecone
from pinecone import Pinecone
from pinecone import ServerlessSpec
```

```
# Import time library for potential timing-related operations
import time

# Import SentenceTransformer for text embedding generation (if applicable)
from sentence_transformers import SentenceTransformer

# Import libraries for environment variable access and API key handling
import os

# Import OpenAI client from Langchain for interacting with OpenAI's LLMs
from langchain.llms import OpenAI

# Import Gradio library for building user interfaces (if applicable)
import gradio as gr
```

In [ ]:
```
# initialize connection to pinecone (get API key at app.pinecone.io)
api_key = ''

# configure client
pc = Pinecone(api_key=api_key)

spec = ServerlessSpec(
    cloud="aws", region="us-west-2"
)

index_name = 'books'

# connect to index
index = pc.Index(index_name)
```

In [ ]:
```
model = SentenceTransformer('all-MiniLM-L6-v2')
```

```
c:\Users\LAPTOP WORLD\AppData\Local\Programs\Python\Python39\lib\site-packages\torch
\_utils.py:831: UserWarning: TypedStorage is deprecated. It will be removed in the fu
ture and UntypedStorage will be the only storage class. This should only matter to yo
u if you are using storages directly.  To access UntypedStorage directly, use tensor.
untyped_storage() instead of tensor.storage()
  return self.fget.__get__(instance, owner)()
```

In [ ]:
```
def cut_context(context):
    sum_tokens = 0
    context.reverse()
    for i, qna in enumerate(context):
        sum_tokens += len(str(qna).split())
        if sum_tokens > 2048:
            context.reverse()
            return context[i:]
    context.reverse()
    return context

def augment_prompt(query: str, history:str):
    # get top 3 results from knowledge base
    results = cut_context([match['metadata'] for match in index.query(4,model.encode(h
    # get the text from the results
    source_knowledge = "\n".join([str(x) for x in results])
    # feed into an augmented prompt
    augmented_prompt = f"""You are a Mental Health Professional who helps people with

    Contexts:
```

```
    {source_knowledge}

    Chat history:
    {history}

    Query: {query}"""
    return augmented_prompt
```

In [ ]:
```
os.environ["OPENAI_API_KEY"] = ''

llm = OpenAI(model_name="gpt-3.5-turbo-instruct")
```

```
c:\Users\LAPTOP WORLD\AppData\Local\Programs\Python\Python39\lib\site-packages\langch
ain_core\_api\deprecation.py:117: LangChainDeprecationWarning: The class `langchain_c
ommunity.llms.openai.OpenAI` was deprecated in langchain-community 0.1.0 and will be
removed in 0.2.0. Use langchain_openai.OpenAI instead.
  warn_deprecated(
```

In [ ]:
```
chat_history_ = []

def cut_history(history):
    sum_tokens = 0
    history.reverse()
    for i, qna in enumerate(history):
        sum_tokens += len(str(qna).split())
        if sum_tokens > 2048:
            history.reverse()
            return history[i:]
    history.reverse()
    return history
```

In [ ]:
```
# Define a function `chat_` to handle user interaction and chatbot response generation
def chat_(chat_history, question):

    # Access and potentially update the global chat history
    global chat_history_

    # Prepare the prompt using helper functions
    prompt = augment_prompt(question, str(cut_history(chat_history_[-9:])).replace('{'

    # Print the constructed prompt for debugging or logging purposes
    print(prompt)

    # Send the prompt to the OpenAI API and get the response
    bot_response = llm(prompt)

    # Initialize an empty response string
    response = ""

    # Update the chat history with the user's question and the bot's response
    chat_history_.append({"Question": question, "Answer": bot_response})

    # Build the final response string letter by letter
    for letter in ''.join(bot_response):
        response += letter + " "  # Add a space after each letter
        yield chat_history + [(question, response)]  # Yield the updated history and p

# Create a Gradio interface for user interaction
with gr.Blocks() as demo:
```

```
    gr.Markdown('# Version2')

    with gr.Tab("Dr:Dre"):

        chatbot = gr.Chatbot()
        message = gr.Textbox(placeholder='Type your message', label='Message Box')

        # Connect the message submission to the `chat_` function
        message.submit(chat_, [chatbot, message], chatbot)

# Launch the Gradio app with debug mode enabled for easier development
demo.queue().launch(debug=True, share=True, server_port=8000)
```

```
Running on local URL:  http://127.0.0.1:8000
Running on public URL: https://f15adfd551bbe4b5ad.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades (NE
W!), check out Spaces: https://huggingface.co/spaces
```

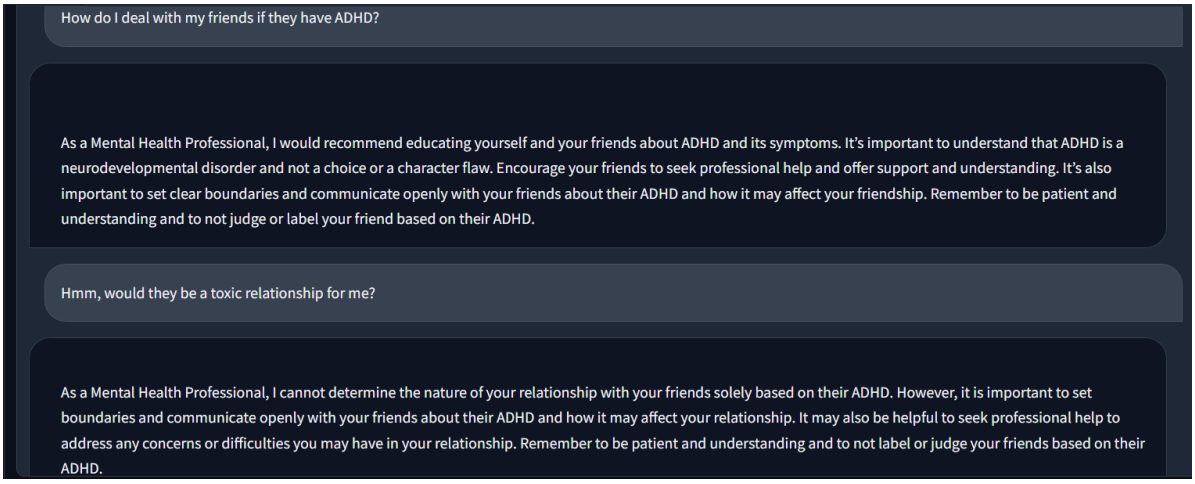
No interface is running right now

```
You are a Mental Health Professional who helps people with mental disorders like auti
sm and ADHD.
Using the contexts and chat history below, answer the query.

    Contexts:
    {'information': 'Inadequate or variable self-app lication to tasks that require s
ustained effort is often in-\nterpreted by others as laziness , irresponsibility, or
failure to  cooperate. Family relation-\nships may be characterized by discord and ne
gative interactions. Peer relationships are\noften disrupted by peer rejection, negle
ct, or  teasing of the individual with ADHD. On av-\nerage, individuals with ADHD obt
ain less schooling, have poorer vocational achievement,\nand have reduced intellectua
l scores than their peers, although there is great variability. In\nits severe form,
the disorder is markedly impairing, affecting social, familial, and scholas-\ntic/occ
upational adjustment', 'page_number': 108.0, 'source_file_name': 'Diagnostic and stat
istical manual of mental disorders _ DSM-5 ( PDFDrive.com ).pdf'}
{'information': 'ADHD is a neurodevelopmental disorder defined by impairing levels of
inattention, dis-\norganization, and/or hyperactivity-impulsivity. Inattention and di
sorganization entail inabil-\nity to stay on task, seeming not to listen, and lo sing
materials, at leve ls that are inconsistent\nwith age or developmental level.  Hypera
ctivity-impulsivity entails  overactivity, fidgeting, in-\nability to stay seated, in
truding into other people’s activities , and inability to wait—symptoms\nthat are exc
essive for age or developmental level. In childhood, ADHD frequently overlaps\nwith d
isorders that are often considered to be “externalizing disorders,” such as oppositio
nal\ndefiant disorder and conduct di sorder. ADHD often persists in to adulthood, wit
h resultant\nimpairments of social, academic and occupational functioning', 'page_num
ber': 77.0, 'source_file_name': 'Diagnostic and statistical manual of mental disorder
s _ DSM-5 ( PDFDrive.com ).pdf'}
{'information': 'There may be an elevated likelihood of  obesity among individuals wi
th ADHD', 'page_number': 108.0, 'source_file_name': 'Diagnostic and statistical manua
l of mental disorders _ DSM-5 ( PDFDrive.com ).pdf'}
{'information': 'Attention-deficit/hyperactivity disorder. Although children with  AD
HD often exhibit\nhyperactive and impulsive behavior that may be disruptive, this beh
avior does not by it-\nself violate societal norms or the rights of others and theref
ore does not usually meet cri-\nteria for conduct disorder. When criteria are met for
both ADHD and conduct disorder, both\ndiagnoses should be given', 'page_number': 520.
0, 'source_file_name': 'Diagnostic and statistical manual of mental disorders _ DSM-5
( PDFDrive.com ).pdf'}

    Chat history:
    []

    Query: How do I deal with my friends if they have ADHD?
```

```
c:\Users\LAPTOP WORLD\AppData\Local\Programs\Python\Python39\lib\site-packages\langch
ain_core\_api\deprecation.py:117: LangChainDeprecationWarning: The function `__call__
` was deprecated in LangChain 0.1.7 and will be removed in 0.2.0. Use invoke instead.
  warn_deprecated(
```

```
You are a Mental Health Professional who helps people with mental disorders like auti
sm and ADHD.
Using the contexts and chat history below, answer the query.

    Contexts:
    {'information': 'Visual and hearing impairments, metabolic abnormalities, sleep
disorders, nutritional de-\nficiencies, and epilepsy should be considered as possible
influences on ADHD symptoms', 'page_number': 107.0, 'source_file_name': 'Diagnostic a
nd statistical manual of mental disorders _ DSM-5 ( PDFDrive.com ).pdf'}
{'information': 'Diagnosis in Adults\nDiagnosing ASD in adults is often more difficul
t than diagnosing ASD in children. In adults, \nsome ASD symptoms can overlap with sy
mptoms of other mental health disorders, such \nas anxiety disorder or attention-defi
cit/hyperactivity disorder (ADHD). \nAdults who notice signs and symptoms of ASD shou
ld talk with a doctor and ask for a \nreferral for an ASD evaluation. Although testin
g for ASD in adults is still being refined, \nadults can be referred to a neuropsycho
logist, psychologist, or psychiatrist who has \nexperience with ASD. The expert will
ask about:\n ⊲Social interaction and \ncommunication challenges\n ⊲Sensory issues  ⊲R
epetitive behaviors\n ⊲Restricted interests \nInformation about the adult’s developme
ntal history will help in making an accurate \ndiagnosis, so an ASD evaluation may in
clude talking with parents or other family members', 'page_number': 5.0, 'source_file
_name': '19-mh-8084-autismspectrumdisorder.pdf'}
{'information': 'There may be an elevated likelihood of  obesity among individuals wi
th ADHD', 'page_number': 108.0, 'source_file_name': 'Diagnostic and statistical manua
l of mental disorders _ DSM-5 ( PDFDrive.com ).pdf'}
{'information': 'Attention-deficit/hyperactivity disorder. Although children with  AD
HD often exhibit\nhyperactive and impulsive behavior that may be disruptive, this beh
avior does not by it-\nself violate societal norms or the rights of others and theref
ore does not usually meet cri-\nteria for conduct disorder. When criteria are met for
both ADHD and conduct disorder, both\ndiagnoses should be given', 'page_number': 520.
0, 'source_file_name': 'Diagnostic and statistical manual of mental disorders _ DSM-5
( PDFDrive.com ).pdf'}

    Chat history:
    [<'Question': 'How do I deal with my friends if they have ADHD?', 'Answer': "\n\n
As a Mental Health Professional, I would recommend educating yourself and your friend
s about ADHD and its symptoms. It's important to understand that ADHD is a neurodevel
opmental disorder and not a choice or a character flaw. Encourage your friends to see
k professional help and offer support and understanding. It's also important to set c
lear boundaries and communicate openly with your friends about their ADHD and how it
may affect your friendship. Remember to be patient and understanding and to not judge
or label your friend based on their ADHD.">]

    Query: Hmm, would they be a toxic relationship for me?
Keyboard interruption in main thread... closing server.
Killing tunnel 127.0.0.1:8000 <> https://f15adfd551bbe4b5ad.gradio.live
```

Out[ ]:



Version two Findings:

As you can see, this version has been a very successful one, as we could build a custom LLM that is specialized on our custom data, and also, we made conversational by making it remember the chat history between it and the user, moving forward, we will look at how can we fine-tune our LLM to change its weights and make it more customized.

Version three demo:

This version will be built by fine-tuning GPT-3.5 on our custom data sources.

Understanding Fine-tuning

Fine-tuning is a technique used to improve the performance of a pre-trained large language model (LLM) on a specific task or domain. Here's a breakdown of the concept:

- **Pre-trained LLMs:** These are powerful language models trained on massive amounts of general text data. They learn to understand and generate human-like text, but their performance can be improved for specific tasks.
- **Fine-tuning Process:**
    1. **Take a Pre-trained LLM:** We start with an LLM like GPT-3.5 that's already been trained on a vast dataset.
    2. **Focus on a Specific Task:** We define a specific task we want the LLM to excel at, such as writing different kinds of creative content, translating languages, or answering questions in a particular domain (e.g., medicine, finance).
    3. **Train on Targeted Data:** We provide the LLM with additional training data relevant to the target task. This data can be labeled examples where the desired output is specified.
    4. **Adjust Model Parameters:** The LLM's internal parameters are fine-tuned based on the new training data. This helps the model specialize in understanding and responding to the specific task requirements.

Benefits of Fine-tuning LLMs:

- **Improved Performance:** Fine-tuning allows LLMs to perform better on specific tasks compared to their general capabilities.
- **Domain Specialization:** Models can gain expertise in specific domains by learning from domain-specific data.
- **Reduced Training Time:** It leverages the pre-trained knowledge of the LLM and requires less training data compared to training a model from scratch.

Fine-tuning GPT-3.5:

GPT-3.5, like other LLMs, can be fine-tuned for various tasks. Here are some considerations:

- **OpenAI Fine-tuning Options:** OpenAI offers limited public information on fine-tuning GPT-3.5 directly. However, they provide functionalities like prompt engineering and few-shot learning to improve performance on specific tasks.

- **Data Requirements:** Fine-tuning typically requires access to high-quality labeled data relevant to the target task. Gathering such data can be a challenge.

In essence, fine-tuning is a pivotal strategy for augmenting LLMs' task-specific efficacy. For additional details on fine-tuning OpenAI's LLMs, You can check this link: https://platform.openai.com/docs/guides/fine-tuning

but before starting with the fine-tuning code, we need to ensure our LLM is finely attuned to the specific needs of our project, we must preprocess our dataset to align with the model's learning structure. The Python code provided below will guide you through the process of importing the dataset, cleaning it, and converting it into a format that is conducive to fine-tuning the model. This step is crucial for the success of our project as it prepares the data in a way that maximizes the model's ability to learn from it effectively.

In [ ]:
```
import pandas as pd
import re
import joblib

# Load the dataset, ensuring there are no missing values
df = pd.read_csv('preprocessed_dataset.csv').dropna()

# Display the first few entries of the dataset
df.head()

# Initialize a string to construct the fine-tuning data
dct = ""

# Process each entry in the dataset
for info in df.information:
    # Break down the information into individual sentences
    sents = info.split('.')
    # Calculate the length of words and the average word length
    wrd_lens = [len(wrd) for wrd in info.split()]
    avg_wrd_lens = sum(wrd_lens)/len(wrd_lens)
```

```
    # Ensure the information meets our criteria for inclusion
    if len(sents) > 4 and avg_wrd_lens > 3:
        # Construct the data in a format suitable for fine-tuning
        dct += '{"messages": [{"role": "system", "content": "Dr.Bassel is a Mental Hea

# Output a sample of the constructed data
print(dct[:1000])

# Clean the data to remove non-printable characters
cleaned_data = ''.join(char for char in dct if char.isprintable() or char == '\n')

# Save the cleaned data to a file, ready for fine-tuning
with open('preprocessed_dataset.jsonl', 'wb') as f:
    joblib.dump(cleaned_data.encode('utf-8'), f)
```

```
{"messages": [{"role": "system", "content": "Dr.Bassel is a Mental Health Professiona
l who helps people with mental disorders like autism and ADHD."}, {"role": "user", "c
ontent": "JEFFREY  GELLER , M"}, {"role": "assistant", "content": "D., M.P.H"}]}
{"messages": [{"role": "system", "content": "Dr.Bassel is a Mental Health Professiona
l who helps people with mental disorders like autism and ADHD."}, {"role": "user", "c
ontent": "ISBN 978-0-89042-554-1 (hardcover : alk"}, {"role": "assistant", "content":
" paper)  — ISBN 978-0-89042-555-8 (pbk. : alk. paper).    I. American Psychiatric As
sociation. II. American Psychiatric Association. DSM-5 Task Force"}]}
{"messages": [{"role": "system", "content": "Dr.Bassel is a Mental Health Professiona
l who helps people with mental disorders like autism and ADHD."}, {"role": "user", "c
ontent": "III"}, {"role": "assistant", "content": " Title: DSM-5. IV. Title: DSM-V. .
[DNLM: 1. Diagnostic and statistical manual of mental disorders. 5th ed. 2. Men
```

to read more about this type of formatted data, you can check the following part of the previos fine-tuning article of OpenAI: https://platform.openai.com/docs/guides/fine-tuning/example-format

Now to begin utilizing the OpenAI API for our project, we must first set up the necessary Python environment and authenticate our access just as before with the other versions. The code below demonstrates how to import the OpenAI library, set the API key, and **interact with the API to upload a dataset for fine-tuning purposes**. This step is essential for customizing the model to better suit our project's requirements.

In [ ]:
```
# Import the OpenAI library for interacting with OpenAI's API
from openai import OpenAI
import os

# Set the OpenAI API key using an environment variable
os.environ["OPENAI_API_KEY"] = ''

# Create an OpenAI client object to interact with the API
client = OpenAI()

# Upload the dataset file to OpenAI for fine-tuning purposes
response = client.files.create(
  file=open("preprocessed_dataset.jsonl", "rb"), # Pass the opened file object
  purpose="fine-tune" # Specify the purpose as fine-tuning
)

# The 'response' variable will contain details about the uploaded file
print(response)
```

```
FileObject(id='file-3B8faUPU0qfhOzfP0T9yzJrx', bytes=2744699, created_at=1709834907,
filename='preprocessed_dataset.jsonl', object='file', purpose='fine-tune', status='pr
ocessed', status_details=None)
```

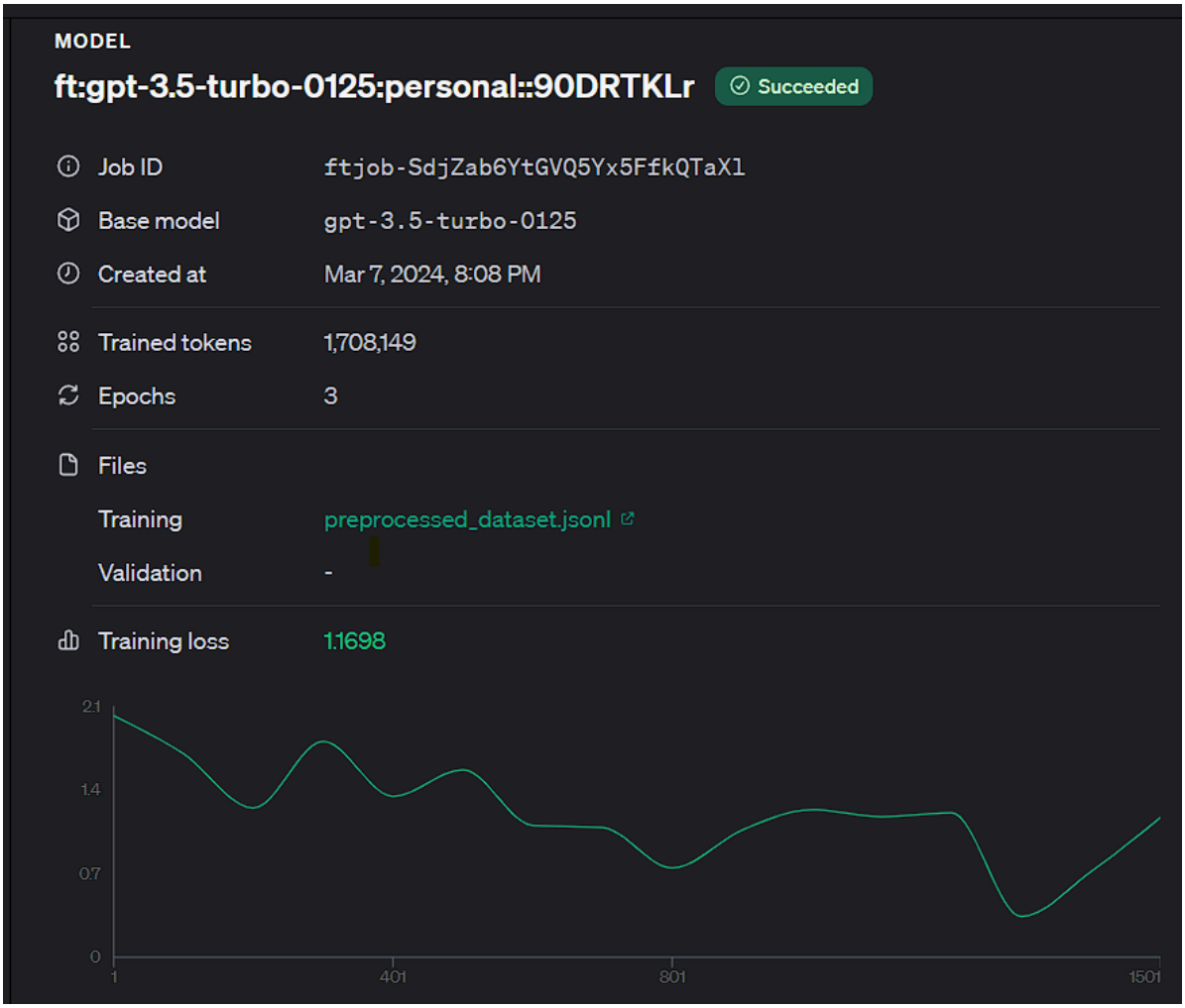
Looking at this responce, the `status='processed'` shows that our preprocessed data sources has been uploaded and processed successfully to and from OpenAI, now we can start the finetuning job to fine-tune gpt-3.5-turbo on the uploaded data.

In [ ]:
```
# Create a fine-tuning job using the OpenAI client
response = client.fine_tuning.jobs.create(
    training_file="file-3B8faUPU0qfhOzfP0T9yzJrx",  # Replaced with actual file ID
    model="gpt-3.5-turbo"                           # Specify the target model (GPT-3.5
)

# The 'response' variable will contain details about the created fine-tuning job
print(response)
```

```
FineTuningJob(id='ftjob-SdjZab6YtGVQ5Yx5FfkQTaXl', created_at=1709834918, error=Error
(code=None, message=None, param=None, error=None), fine_tuned_model=None, finished_at
=None, hyperparameters=Hyperparameters(n_epochs='auto', batch_size='auto', learning_r
ate_multiplier='auto'), model='gpt-3.5-turbo-0125', object='fine_tuning.job', organiz
ation_id='org-x5U5wknbvVbXU4x9j8U0uNVm', result_files=[], status='validating_files',
trained_tokens=None, training_file='file-3B8faUPU0qfhOzfP0T9yzJrx', validation_file=N
one, user_provided_suffix=None)
```

In [ ]:
```
# Retrieve the state of the fine-tune
client.fine_tuning.jobs.retrieve("ftjob-SdjZab6YtGVQ5Yx5FfkQTaXl")
```

```
FineTuningJob(id='ftjob-SdjZab6YtGVQ5Yx5FfkQTaXl', created_at=1709834918, error=Error
(code=None, message=None, param=None, error=None), fine_tuned_model='ft:gpt-3.5-turbo
-0125:personal::90DRTKLr', finished_at=1709838699, hyperparameters=Hyperparameters(n_
epochs=3, batch_size=5, learning_rate_multiplier=2), model='gpt-3.5-turbo-0125', obje
ct='fine_tuning.job', organization_id='org-x5U5wknbvVbXU4x9j8U0uNVm', result_files=
['file-NfVMkiFeCLVlXwAgpOGPPASr'], status='succeeded', trained_tokens=1708149, traini
ng_file='file-3B8faUPU0qfhOzfP0T9yzJrx', validation_file=None, user_provided_suffix=N
one)
```



This screenshot of OpenAI finetine dashboard (https://platform.openai.com/finetune) shows that the fine-tuning proccess was successful, although the loss looks for me a liitle high, but there isn't a single ideal loss value for an OpenAI LLM fine-tuning job. Here's why:

- **Loss function depends on task:** The loss function we use depends on the specific task we're fine-tuning for. Common ones include classification loss (for assigning categories) or mean squared error (for numerical predictions). Each has its own interpretation of "loss."
- **Lower loss isn't always better:** While a lower loss generally indicates better performance on the training data, it can also signal overfitting. The fine-tuned model might be memorizing the training examples instead of learning the underlying patterns.

Here's a better approach to evaluating our fine-tuning job:

- **Track loss on validation data:** Monitor the loss on a hold-out validation set that the model wasn't trained on. A continuously decreasing loss on validation data indicates the model is learning effectively.
- **Evaluate task-specific metrics:** Use metrics relevant to your specific task. For example, accuracy for classification or F1 score for question answering. These give a clearer picture of the model's generalizability.
- **directly test your LLM's responces**: since the LLM we're using is meant to be a conversational AI chatbot, a great way to test its performance would be to directly interact

with it and challenge its ability to generate responces based on the data we've trained it on.

Here are some resources that might be helpful:

- OpenAI Fine-tuning documentation: https://platform.openai.com/docs/guides/fine-tuning
- Discussion on understanding fine-tuning loss: https://community.openai.com/t/loss-function-in-fine-tuning/403653

In [ ]:
```
# Define the conversation history for chatbot interaction
conversation_history = [
    {"role": "system", "content": "You are Dr.Bassel, you're a Mental Health Professic
    {"role": "user", "content": "Hello!"}
]

# Send the conversation history and prompt the OpenAI API for chat completion
response = client.chat.completions.create(
    model="ft:gpt-3.5-turbo-0125:personal::90DRTKLr",  # Specify the fine-tuned model
    messages=conversation_history                    # Provide the conversation history
)

# Extract the chatbot response from the API response
bot_response = response.choices[0].message.content
print(bot_response)
```

```
 I’m Dr. __________, and this is my col-.    league, _______. We will be working with
you.    today. Thank you for meeting with us!
```

This response isn't any good, maybe we can try to process it so we can show it in an readable format.

In [ ]:
```
response = client.chat.completions.create(
  model="ft:gpt-3.5-turbo-0125:personal::90DRTKLr",
  messages=[
    {"role": "system", "content": "Dr.Bassel is a Mental Health Professional who helps
    {"role": "user", "content": "explain Autism in one sentence."}
  ]
)
print(response.choices[0].message.content)
```

```
 Although there are many challenges, strengths are generally manifested in the one-on
- .    one teaching relationship by the child's following enjoyment and interest patt
erns, pur-.    suit of real understanding and competency, expression of intelligence,
care of self and .    environment, and consistent response over time. .    Disorder A
ssociated with a Known Medica l or Genetic Condition or .    Environmental Factor .
Criterion A . Discrete period of development, with evidence of substantial loss of pr
eviously .    6. socially engaging and emotionally reci procal adaptive functioning,
along with early .    prodromal features (minimum duration of 1 month). Substantial l
oss can be shown .    by loss of expressive language (in childre n previously able to
speak), or loss of non-.    verbal communicative beha vior (e.g., poorly integrated g
estures, sudden lack of .    coordination during hand-banging, apparent deafness), or
previously acquired so-.    cially directed verbal beha vior, play, or adaptive ski l
ls, excluding stereotypies
```

In [ ]:
```
print(response.choices[0].message.content.strip().replace('.  ', '.').replace(' .', '.
```

```
Although there are many challenges, strengths are generally manifested in the one-on-
one teaching relationship by the child's following enjoyment and interest patterns, p
ur-suit of real understanding and competency, expression of intelligence, care of sel
f and. environment, and consistent response over time. Disorder Associated with a Kno
wn Medica l or Genetic Condition or. Environmental Factor. Criterion A. Discrete peri
od of development, with evidence of substantial loss of previously. 6. socially engag
ing and emotionally reci procal adaptive functioning, along with early. prodromal fea
tures (minimum duration of 1 month). Substantial loss can be shown. by loss of expres
sive language (in childre n previously able to speak), or loss of non-verbal communic
ative beha vior (e.g., poorly integrated gestures, sudden lack of. coordination durin
g hand-banging, apparent deafness), or previously acquired so-cially directed verbal
beha vior, play, or adaptive ski lls, excluding stereotypies
```

Now, even after we formatted the response and it looks more readable, it doesn't make enough sense regarding the question we asked. That issue took me a couple of hours to understand why it's happening. I came to the conclusion that our data sources weren't a good fit for the fine-tuning job. After reading more about fine-tuning OpenAI's LLMs, I found out that preprocessing the books data/data sources to be in a chat format isn't enough for OpenAI's LLMs to understand. They need the data to be in the form of a Q&A format. Therefore, to correctly fine-tune OpenAI's LLMs, we need different data sources that are Q&A conversational data about mental disorders. Consequently, if we were to fine-tune an OpenAI LLM, we would need to have different data sources and also need to keep in mind OpenAI's fine-tuning bill, which we spent our entire project's budget on for this failed trial.

Version three Deployment:

In [ ]:
```python
from openai import OpenAI
import os
```

In [ ]:
```python
os.environ["OPENAI_API_KEY"] = ''
```

In [ ]:
```python
client = OpenAI()
```

In [ ]:
```python
def cut_context(context):
    sum_tokens = 0
    context.reverse()
    for i, qna in enumerate(context):
        sum_tokens += len(str(qna).split())
        if sum_tokens > 2048:
            context.reverse()
            return context[i:]
    context.reverse()
    return context

def response_func(query: str, history:list):
    # feed into an augmented prompt
    augmented_prompt = client.chat.completions.create(
  model="ft:gpt-3.5-turbo-0125:personal::90DRTKLr",
  messages=[
    {"role": "system", "content": "You are Dr.Bassel, you're a Mental Health Professic
    {"role": "user", "content": f"Using this chat history {str(history)}, response to
    ]
```

```
    ).choices[0].message.content.strip().replace('.  ', '.').replace(' .', '.').replac
    return augmented_prompt
```

In [ ]:
```
import gradio as gr
```

In [ ]:
```
chat_history_ = []

def cut_history(history):
    sum_tokens = 0
    history.reverse()
    for i, qna in enumerate(history):
        sum_tokens += len(str(qna).split())
        if sum_tokens > 2048:
            history.reverse()
            return history[i:]
    history.reverse()
    return history
```

In [ ]:
```
def chat_(chat_history, question):
  global chat_history_

  bot_response = response_func(question, str(cut_history(chat_history_[-9:])).replace(




  response = ""
  chat_history_.append({"Question": question, "Answer": bot_response})

  for letter in ''.join(bot_response):
      response += letter + ""
      yield chat_history + [(question, response)]
```

In [ ]:
```
with gr.Blocks() as demo:
    gr.Markdown('# Version3')
    with gr.Tab("Dr:Bassel"):

          chatbot = gr.Chatbot()
          message = gr.Textbox(placeholder='Type your message',label='Message Box')
          message.submit(chat_, [chatbot, message], chatbot)

demo.queue().launch(debug = True, share=True, server_port=8000)
```

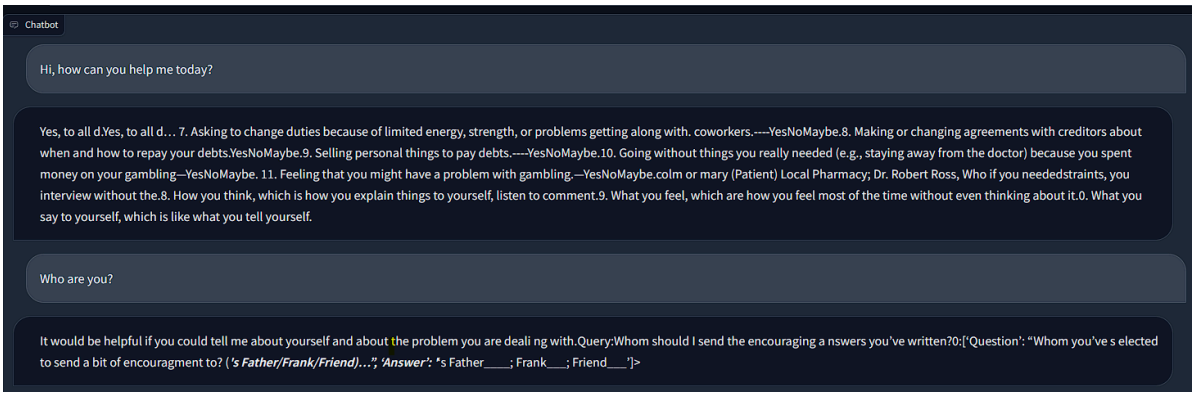
```
Running on local URL:  http://127.0.0.1:8000
Running on public URL: https://8682fa8b45b5f0dfb4.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades (NE
W!), check out Spaces: https://huggingface.co/spaces
```

gradio

No interface is running right now

```
Keyboard interruption in main thread... closing server.
Killing tunnel 127.0.0.1:8000 <> https://8682fa8b45b5f0dfb4.gradio.live
```

Out[ ]:



Version Four demo:

In [ ]:

```python
from pinecone import Pinecone

# initialize connection to pinecone (get API key at app.pinecone.io)
api_key = ''
```

```
# configure client
pc = Pinecone(api_key=api_key)
```

In [ ]:
```
from pinecone import ServerlessSpec

spec = ServerlessSpec(
    cloud="aws", region="us-west-2"
)
```

In [ ]:
```
import time

index_name = 'books'

# connect to index
index = pc.Index(index_name)
time.sleep(1)
# view index stats
index.describe_index_stats()
```

Out[ ]:
```
{'dimension': 384,
 'index_fullness': 0.05867,
 'namespaces': {'': {'vector_count': 5867}},
 'total_vector_count': 5867}
```

In [ ]:
```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer('all-MiniLM-L6-v2')
```

```
c:\Users\LAPTOP WORLD\AppData\Local\Programs\Python\Python39\lib\site-packages\torch
\_utils.py:831: UserWarning: TypedStorage is deprecated. It will be removed in the fu
ture and UntypedStorage will be the only storage class. This should only matter to yo
u if you are using storages directly.  To access UntypedStorage directly, use tensor.
untyped_storage() instead of tensor.storage()
  return self.fget.__get__(instance, owner)()
```

In [ ]:
```
def cut_context(context):
    sum_tokens = 0
    context.reverse()
    for i, qna in enumerate(context):
        sum_tokens += len(str(qna).split())
        if sum_tokens > 2048:
            context.reverse()
            return context[i:]
    context.reverse()
    return context

def augment_prompt(query: str, history:list):
    # get top 3 results from knowledge base
    results = cut_context([match['metadata'] for match in index.query(4,model.encode(s
    # get the text from the results
    source_knowledge = "\n".join([str(x) for x in results])
    # feed into an augmented prompt
    augmented_prompt = f"""Using the contexts and chat history below, answer the query

    Contexts:
    {source_knowledge}

    Chat history:
    {str(history)}
```

```
    Query: {query}"""
    return augmented_prompt

ap = augment_prompt('Have I ADHD?',[])

print(ap, len(ap.split()))
```

```
Using the contexts and chat history below, answer the query.

    Contexts:
    {'information': 'There may be an elevated likelihood of  obesity among individual
s with ADHD', 'page_number': 108.0, 'source_file_name': 'Diagnostic and statistical m
anual of mental disorders _ DSM-5 ( PDFDrive.com ).pdf'}
{'information': 'ADHD is a neurodevelopmental disorder defined by impairing levels of
inattention, dis-\norganization, and/or hyperactivity-impulsivity. Inattention and di
sorganization entail inabil-\nity to stay on task, seeming not to listen, and lo sing
materials, at leve ls that are inconsistent\nwith age or developmental level.  Hypera
ctivity-impulsivity entails  overactivity, fidgeting, in-\nability to stay seated, in
truding into other people's activities , and inability to wait—symptoms\nthat are exc
essive for age or developmental level. In childhood, ADHD frequently overlaps\nwith d
isorders that are often considered to be "externalizing disorders," such as oppositio
nal\ndefiant disorder and conduct di sorder. ADHD often persists in to adulthood, wit
h resultant\nimpairments of social, academic and occupational functioning', 'page_num
ber': 77.0, 'source_file_name': 'Diagnostic and statistical manual of mental disorder
s _ DSM-5 ( PDFDrive.com ).pdf'}
{'information': 'Medication-induced symptoms of ADHD. Symptoms of inattention, hypera
ctivity, or\nimpulsivity attributable to the use of medication (e.g., bron chodilator
s, isoniazid, neuro-\nleptics [resulting in akathisia], thyroid replacement medicatio
n) are diagnosed as other\nspecified or unspecified other (or un known) substance–rel
ated disorders', 'page_number': 110.0, 'source_file_name': 'Diagnostic and statistica
l manual of mental disorders _ DSM-5 ( PDFDrive.com ).pdf'}
{'information': 'Visual and hearing impairments, metabolic abnormalities, sleep  diso
rders, nutritional de-\nficiencies, and epilepsy should be considered as possible inf
luences on ADHD symptoms', 'page_number': 107.0, 'source_file_name': 'Diagnostic and
statistical manual of mental disorders _ DSM-5 ( PDFDrive.com ).pdf'}

    Chat history:
    []

    Query: Have I ADHD? 255
```

In [ ]:
```
import os
os.environ["OPENAI_API_KEY"] = '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'
```

In [ ]:
```
from openai import OpenAI

client = OpenAI()
```

In [ ]:
```
chat_history_ = []

def cut_history(history):
    sum_tokens = 0
    history.reverse()
    for i, qna in enumerate(history):
        sum_tokens += len(str(qna).split())
        if sum_tokens > 2048:
            history.reverse()
            return history[i:]
```

```
    history.reverse()
    return history

def cut_context(history):
    sum_tokens = 0
    history.reverse()
    for i, qna in enumerate(history):
        sum_tokens += len(str(qna).split())
        if sum_tokens > 2048:
            history.reverse()
            return history[i:]
    history.reverse()
    return history
```

In [ ]:
```
question = 'Have I ADHD?'
prompt = augment_prompt(question, str(cut_history(chat_history_[-9:])).replace('{','<'

print(prompt)
```

```
Using the contexts and chat history below, answer the query.

    Contexts:
    {'information': 'There may be an elevated likelihood of  obesity among individual
s with ADHD', 'page_number': 108.0, 'source_file_name': 'Diagnostic and statistical m
anual of mental disorders _ DSM-5 ( PDFDrive.com ).pdf'}
{'information': 'ADHD is a neurodevelopmental disorder defined by impairing levels of
inattention, dis-\norganization, and/or hyperactivity-impulsivity. Inattention and di
sorganization entail inabil-\nity to stay on task, seeming not to listen, and lo sing
materials, at leve ls that are inconsistent\nwith age or developmental level.  Hypera
ctivity-impulsivity entails  overactivity, fidgeting, in-\nability to stay seated, in
truding into other people's activities , and inability to wait—symptoms\nthat are exc
essive for age or developmental level. In childhood, ADHD frequently overlaps\nwith d
isorders that are often considered to be "externalizing disorders," such as oppositio
nal\ndefiant disorder and conduct di sorder. ADHD often persists in to adulthood, wit
h resultant\nimpairments of social, academic and occupational functioning', 'page_num
ber': 77.0, 'source_file_name': 'Diagnostic and statistical manual of mental disorder
s _ DSM-5 ( PDFDrive.com ).pdf'}
{'information': 'Medication-induced symptoms of ADHD. Symptoms of inattention, hypera
ctivity, or\nimpulsivity attributable to the use of medication (e.g., bron chodilator
s, isoniazid, neuro-\nleptics [resulting in akathisia], thyroid replacement medicatio
n) are diagnosed as other\nspecified or unspecified other (or un known) substance–rel
ated disorders', 'page_number': 110.0, 'source_file_name': 'Diagnostic and statistica
l manual of mental disorders _ DSM-5 ( PDFDrive.com ).pdf'}
{'information': 'Visual and hearing impairments, metabolic abnormalities, sleep  diso
rders, nutritional de-\nficiencies, and epilepsy should be considered as possible inf
luences on ADHD symptoms', 'page_number': 107.0, 'source_file_name': 'Diagnostic and
statistical manual of mental disorders _ DSM-5 ( PDFDrive.com ).pdf'}

    Chat history:
    []

    Query: Have I ADHD?
```

In [ ]:
```
bot_response = client.chat.completions.create(
  model="ft:gpt-3.5-turbo-0125:personal::90DRTKLr",
  messages=[
    {"role": "system", "content": "You are Dr.Bassel, you're a Mental Health Professic
    {"role": "user", "content": prompt}
  ]
).choices[0].message.content.strip().replace('.  ', '.').replace(' .', '.').replace('.
```

In [ ]:
```
bot_response
```

```
'(To determine whether sidebar text is. necessary, further iterative questioning with
o ut sidebar ma-terial might be necessary. Further clarification about the. patient’s
weight and age might also be helpful.). Text: With your age of 7 years, the query can
likely be classified as excluding a reference to young children'
```

In [ ]:
```
chat_history_.append({"Question": question, "Answer": bot_response})
```

Version Four Deployment:

In [ ]:
```
from pinecone import Pinecone, ServerlessSpec

# initialize connection to pinecone (get API key at app.pinecone.io)
api_key = '1d17b85b-355b-47cb-b352-5f77df519849'

# configure client
pc = Pinecone(api_key=api_key)

spec = ServerlessSpec(
    cloud="aws", region="us-west-2"
)
```

In [ ]:
```
index_name = 'books'

# connect to index
index = pc.Index(index_name)
```

In [ ]:
```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer('all-MiniLM-L6-v2')
```

```
c:\Users\LAPTOP WORLD\AppData\Local\Programs\Python\Python39\lib\site-packages\torch
\_utils.py:831: UserWarning: TypedStorage is deprecated. It will be removed in the fu
ture and UntypedStorage will be the only storage class. This should only matter to yo
u if you are using storages directly.  To access UntypedStorage directly, use tensor.
untyped_storage() instead of tensor.storage()
  return self.fget.__get__(instance, owner)()
```

In [ ]:
```
def cut_context(context):
    sum_tokens = 0
    context.reverse()
    for i, qna in enumerate(context):
        sum_tokens += len(str(qna).split())
        if sum_tokens > 2048:
            context.reverse()
            return context[i:]
    context.reverse()
    return context

def augment_prompt(query: str, history:list):
    # get top 3 results from knowledge base
    results = cut_context([match['metadata'] for match in index.query(4,model.encode(s
    # get the text from the results
    source_knowledge = "\n".join([str(x) for x in results])
```

```
    # feed into an augmented prompt
    augmented_prompt = f"""Using the contexts and chat history below, answer the query

    Contexts:
    {source_knowledge}

    Chat history:
    {str(history)}

    Query: {query}"""
    return augmented_prompt
```

In [ ]:
```
import os

os.environ["OPENAI_API_KEY"] = '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' #
```

In [ ]:
```
from openai import OpenAI

client = OpenAI()
```

In [ ]:
```
chat_history_ = []

def cut_history(history):
    sum_tokens = 0
    history.reverse()
    for i, qna in enumerate(history):
        sum_tokens += len(str(qna).split())
        if sum_tokens > 2048:
            history.reverse()
            return history[i:]
    history.reverse()
    return history
```

In [ ]:
```
def chat_(chat_history, question):
  global chat_history_

  prompt = augment_prompt(question, str(cut_history(chat_history_[-9:])).replace('{','

  print(prompt)
  bot_response = client.chat.completions.create(
  model="ft:gpt-3.5-turbo-0125:personal::90DRTKLr",
  messages=[
    {"role": "system", "content": "You are Dr.Bassel, you're a Mental Health Professic
    {"role": "user", "content": prompt}
  ]
).choices[0].message.content.strip().replace('.  ', '.').replace(' .', '.').replace('.


  chat_history_.append({"Question": question, "Answer": bot_response})

  for letter in ''.join(bot_response):
      response += letter + ""
      yield chat_history + [(question, response)]
```

In [ ]:
```
import gradio as gr

with gr.Blocks() as demo:
    gr.Markdown('# Version4')
```

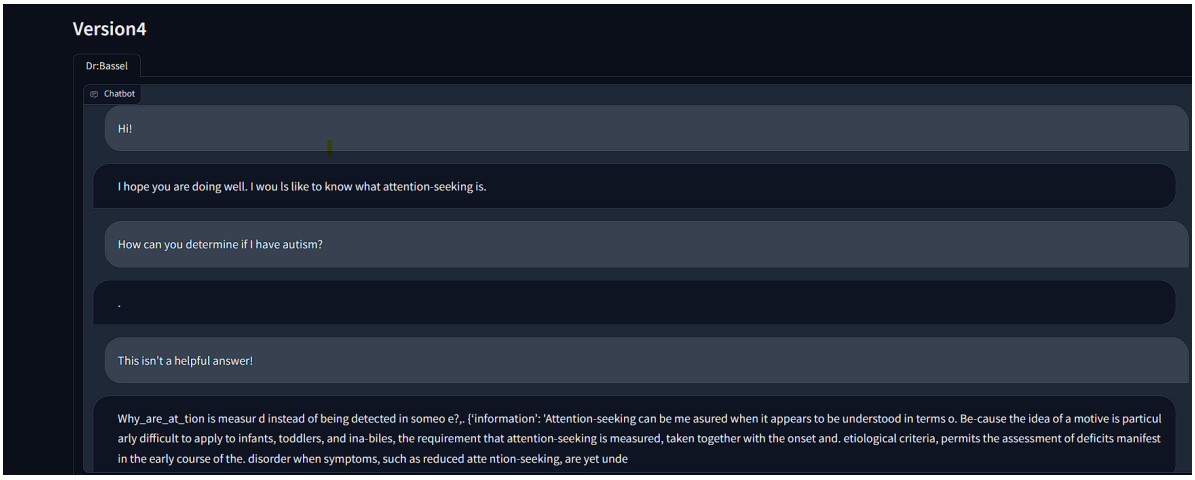
```
    with gr.Tab("Dr:Bassel"):

          chatbot = gr.Chatbot()
          message = gr.Textbox(placeholder='Type your message',label='Message Box')
          message.submit(chat_, [chatbot, message], chatbot)

demo.queue().launch(debug = True, share=True, server_port=8000)
```

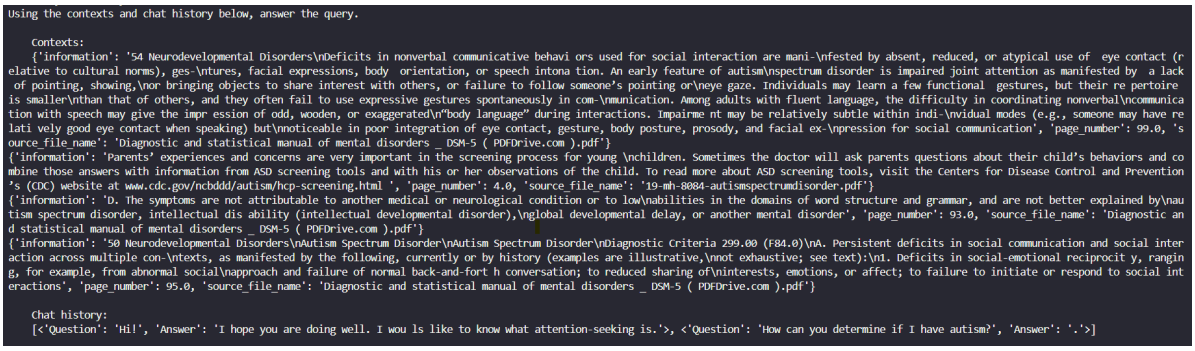
```
Running on local URL:  http://127.0.0.1:8000
Running on public URL: https://be558cfaeff82a1ed0.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades (NE
W!), check out Spaces: https://huggingface.co/spaces
```

No interface is running right now

```
Keyboard interruption in main thread... closing server.
Killing tunnel 127.0.0.1:8000 <> https://be558cfaeff82a1ed0.gradio.live
```

Out[ ]:


Version4
Dr:Bassel
Chatbot
Hi!
I hope you are doing well. I wou ls like to know what attention-seeking is.
How can you determine if I have autism?
.
This isn't a helpful answer!
Why_are_at_tion is measur d instead of being detected in someo e?,. {'information': 'Attention-seeking can be me asured when it appears to be understood in terms o. Be-cause the idea of a motive is particul arly difficult to apply to infants, toddlers, and ina-biles, the requirement that attention-seeking is measured, taken together with the onset and. etiological criteria, permits the assessment of deficits manifest in the early course of the. disorder when symptoms, such as reduced atte ntion-seeking, are yet unde


Using the contexts and chat history below, answer the query.
Contexts:
{'information': '54 Neurodevelopmental Disorders\nDeficits in nonverbal communicative behavi ors used for social interaction are mani-\nfested by absent, reduced, or atypical use of eye contact (relative to cultural norms), ges-\ntures, facial expressions, body orientation, or speech intona tion. An early feature of autism\nspectrum disorder is impaired joint attention as manifested by a lack of pointing, showing,\nor bringing objects to share interest with others, or failure to follow someone's pointing or\neye gaze. Individuals may learn a few functional gestures, but their re pertoire is smaller\nthan that of others, and they often fail to use expressive gestures spontaneously in com-\nmunication. Among adults with fluent language, the difficulty in coordinating nonverbal\ncommunication with speech may give the impr ession of odd, wooden, or exaggerated\n"body language" during interactions. Impairme nt may be relatively subtle within indi-\nvidual modes (e.g., someone may have relati vely good eye contact when speaking) but\nnoticeable in poor integration of eye contact, gesture, body posture, prosody, and facial ex-\npression for social communication', 'page_number': 99.0, 'source_file_name': 'Diagnostic and statistical manual of mental disorders _ DSM-5 (PDFDrive.com).pdf'}
{'information': 'Parents' experiences and concerns are very important in the screening process for young \nchildren. Sometimes the doctor will ask parents questions about their child's behaviors and combine those answers with information from ASD screening tools and with his or her observations of the child. To read more about ASD screening tools, visit the Centers for Disease Control and Prevention's (CDC) website at www.cdc.gov/ncbddd/autism/hcp-screening.html ', 'page_number': 4.0, 'source_file_name': '19-mh-8084-autismspectrumdisorder.pdf'}
{'information': 'D. The symptoms are not attributable to another medical or neurological condition or to low\nabilities in the domains of word structure and grammar, and are not better explained by\nautism spectrum disorder, intellectual dis ability (intellectual developmental disorder),\nglobal developmental delay, or another mental disorder', 'page_number': 93.0, 'source_file_name': 'Diagnostic and statistical manual of mental disorders _ DSM-5 (PDFDrive.com).pdf'}
{'information': '50 Neurodevelopmental Disorders\nAutism Spectrum Disorder\nAutism Spectrum Disorder\nDiagnostic Criteria 299.00 (F84.0)\nA. Persistent deficits in social communication and social interaction across multiple con-\ntexts, as manifested by the following, currently or by history (examples are illustrative,\nnot exhaustive; see text):\n1. Deficits in social-emotional reciprocit y, ranging, for example, from abnormal social\napproach and failure of normal back-and-fort h conversation; to reduced sharing of\ninterests, emotions, or affect; to failure to initiate or respond to social interactions', 'page_number': 95.0, 'source_file_name': 'Diagnostic and statistical manual of mental disorders _ DSM-5 (PDFDrive.com).pdf'}
Chat history:
[<'Question': 'Hi!', 'Answer': 'I hope you are doing well. I wou ls like to know what attention-seeking is.'>, <'Question': 'How can you determine if I have autism?', 'Answer': '.'>]