## **Pre-processing**

pre-processing steps :

  1. **Duplicate Removal**: Identify and remove any duplicate comments in the data to ensure that each comment is unique and representative of the data set.

2. **Translation Removal**: Delete any text segments that are enclosed in parentheses and start with "Translate by Google", as they may contain inaccurate or irrelevant translations that could affect the sentiment analysis results.

3. **Original Text Removal**: Delete any text segments that are enclosed in parentheses and start with "Original", as well as any subsequent text on the same line, as they may contain the original text in a different language that is not relevant for the sentiment analysis task.

4. **Emoji Extraction and Replacement**: Extract all the emojis from each comment and store them in a separate column in the data. Replace the emojis in the comment with their corresponding names or descriptions to capture their semantic meaning. For example, 😊 would be replaced with "smiling face with smiling eyes".

5. **Emoji Sentiment Scoring**: Assign a sentiment score to each emoji based on its polarity using a sentiment intensity analyzer, which is a tool that measures how positive or negative an emoji is on a scale from -1 to 1. Store the sentiment score in another column in the data. For example, 😊 might have a score of 0.8, while 😡 might have a score of -0.9.

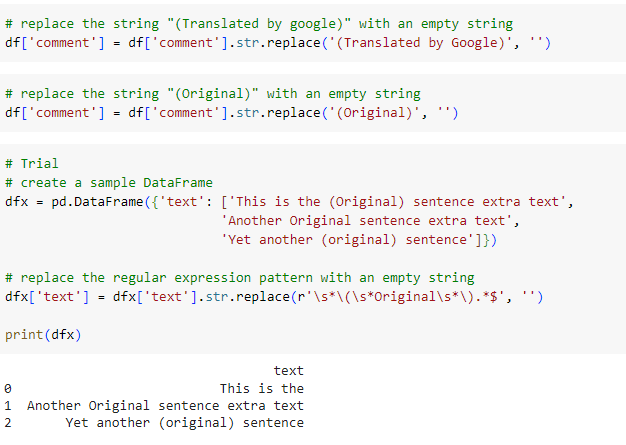
6. **Irrelevant Text Removal**: Remove any text that contains html tags, web links, or other irrelevant information that may not contribute to the sentiment of the comment. This can help simplify and standardize the data and make it more consistent and readable.

7. **Chat Word Expansion**: Expand any abbreviations, acronyms, or slang words that are commonly used in online chats with their full forms or meanings. This can help normalize and clarify the language and make it easier to understand and analyze.

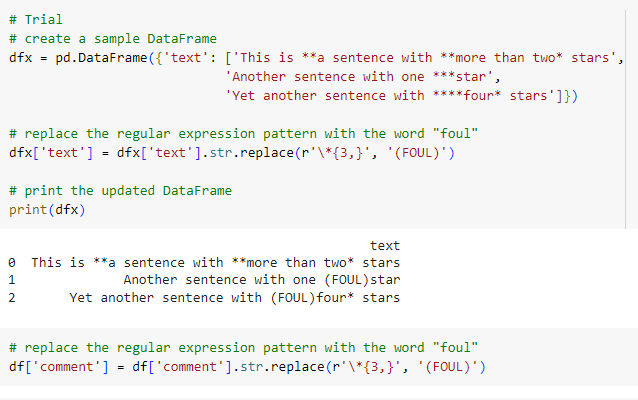
8. **Spell Check**: Correct any spelling errors or typos that may appear in the comments. This can help improve the quality and accuracy of the data and avoid confusion or misinterpretation.

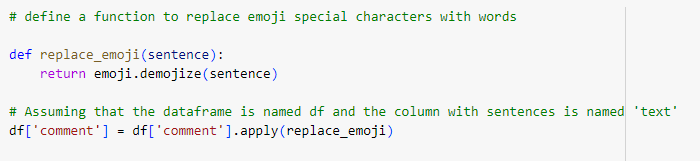
9. **Twitter Handle and Symbol Removal**: Remove any text that starts with @ or #, which are used to mention other users or topics on Twitter, as well as any other symbols or characters that may not convey any meaning or sentiment. This can help focus on the actual content and tone of the comment and remove any distractions or noise.

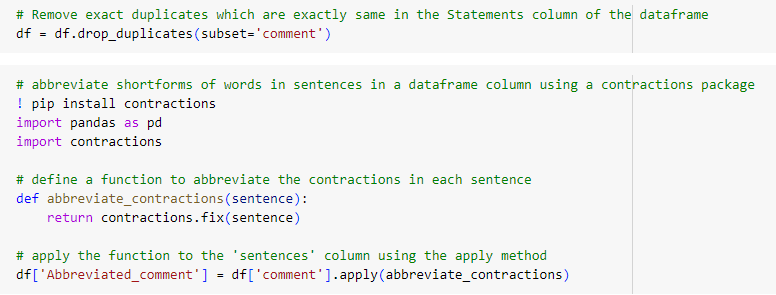
10. **Lower Case Conversion**: Convert all the letters in the comments to lower case after applying all the previous steps. This can help normalize the data and make it easier to convert into numerical vectors or embeddings, which are representations of words or sentences that can be used for machine learning models.











# **Feature Engineering**

Features added :

1. **Emoji Polarity Score**: This is the same as the emoji sentiment score that was mentioned in the previous steps. It is a numerical score for each emoji based on how positive or negative it is on a scale from -1 to 1. The score can be obtained from a sentiment intensity analyzer, which is a tool that assigns a value to an emoji based on its polarity. For example, 😊 might have a score of 0.8, while 😡 might have a score of -0.9. This can help quantify the emotional impact of the emoji on the comment.

2. **Count of Capital Letter Words Used:** This is a feature that counts how many words in the comment are written in all capital letters. This can indicate the intensity or emphasis of the comment, as well as the tone or mood of the reviewer. For example, "I LOVE THIS PRODUCT" would have a count of 4, while "I love this product" would have a count of 0. This can help measure the strength or weakness of the sentiment expressed in the comment.

3. **Yelling Alert to Check if the Reviewer is Yelling**: This is a feature that checks if the comment contains any words that are written in all capital letters and are longer than 3 letters. This can indicate that the reviewer is yelling or shouting, which can imply anger, frustration, or excitement. For example, "THIS IS THE WORST PRODUCT EVER" would trigger a yelling alert, while "This is the worst product ever" would not. This can help identify extreme or exaggerated sentiments in the comment.

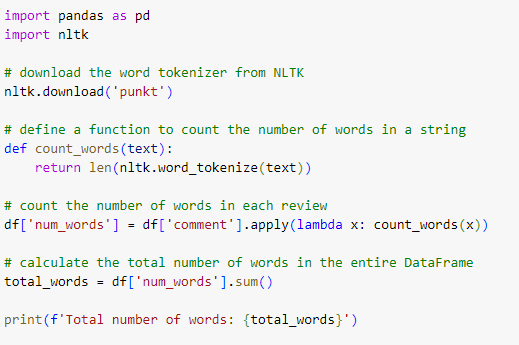
4. **Replacing \* with Bad**: This is a feature that replaces any asterisks (\*) in the comment with the word "bad". This can help capture the negative sentiment of the comment more accurately, as asterisks are often used to censor profanity or offensive words. For example, "This product is sh\*t" would be replaced with "This product is bad". This can help avoid missing or misinterpreting the sentiment of the comment due to censorship.

5. **Added a Profanity Score to Measure the Degree of Profanity in the Review**: This is a feature that assigns a numerical score to each comment based on how many profane or offensive words it contains. The score can be obtained from a profanity detector, which is a tool that identifies and counts profane words in a text. For example, "This product is sh\*t" might have a score of 1, while "This product is awesome" might have a score of 0. This can help measure the degree of profanity in the comment, which can affect the sentiment analysis results.

6. **Added Additional Sentiment Using a Pretrained Model** (ex: sia). Add Polarity and Subjectivity of the Sentence as New Columns: This is a feature that uses a pretrained model (such as sia) to analyze the sentiment of each comment and add two new columns to the data: polarity and subjectivity. Polarity is a measure of how positive or negative a comment is on a scale from -1 to 1, while subjectivity is a measure of how objective or subjective a comment is on a scale from 0 to 1. For example, "This product is awesome" might have a polarity of 0.8 and a subjectivity of 0.9, while "This product works well" might have a polarity of 0.5 and a subjectivity of 0.2. This can help provide additional information and perspectives on the sentiment of the comment.

7. **Length of the Review** (Number of Characters): This is a feature that counts how many characters (including spaces and punctuation) are in each comment. This can indicate how detailed or concise the comment is, as well as how much effort or interest the reviewer has put into writing it. For example, "This product is awesome" would have a length of 20 characters, while "I bought this product last week and I am very happy with it. It works well and it looks nice." would have a length of 67 characters. This can help measure the quality and quantity of the comment.

8. **Number of Words**: This is a feature that counts how many words are in each comment. This can also indicate how detailed or concise the comment is, as well as how much information or opinion it contains. For example, "This product is awesome" would have a number of words of 4, while "I bought this product last week and I am very happy with it. It works well and it looks nice." would have a number of words of 14. This can help measure the complexity and richness of the comment.

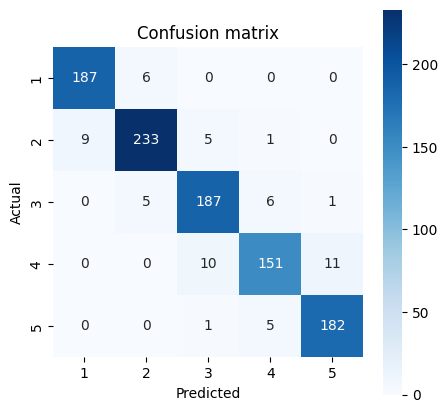


## **Text-Curie-003 Model**

Text Curie 3 is a large language model (LLM) in the OPENAI GPT-3 family that can understand and generate natural language¹. It is named after Marie Curie, the first woman to win the Nobel Prize in Physics and Chemistry². Text Curie is more capable than Text Babbage, but less capable than Text Davinci¹.

The Text Curie 3 model that has been pre-trained on a sizable dataset of text and code. It can be used for a variety of tasks, including sentiment classification, summarization, and question answering.

## **Confusion Matrix for Test Data**



Classification report:

precision recall f1-score support

1 0.95 0.97 0.96 193

2 0.95 0.94 0.95 248

3 0.92 0.94 0.93 199

4 0.93 0.88 0.90 172

5 0.94 0.97 0.95 188

accuracy 0.94 1000

macro avg 0.94 0.94 0.94 1000

weighted avg 0.94 0.94 0.94 1000

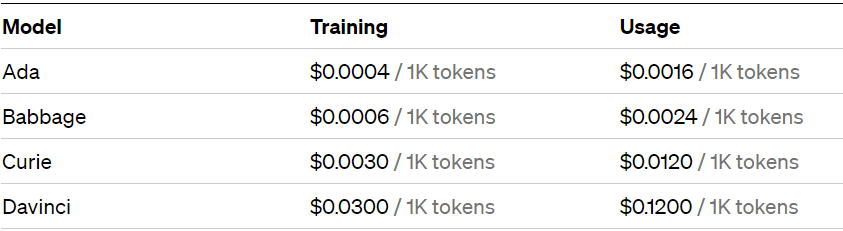
## **What can it do?**

* Text Curie can perform various natural language tasks, such as language translation, complex classification, text sentiment, and summarization¹. It is also good at answering questions and performing Q&A and as a general service chatbot¹.
* Text Curie is based on a deep neural network architecture called Transformer³, which uses attention mechanisms to learn the relationships between words and sentences. Text Curie is trained on a large corpus of text data from various sources, such as books, news articles, web pages, and social media posts⁴. Text Curie learns from the patterns and structures of natural language and can generate coherent and fluent texts based on a given prompt or context.

## **Merits**

* It is very large and has been trained on a massive dataset. This means that it has a deep understanding of language and can generate very accurate and informative text. This allows it to learn the nuances of human language and to identify sentiment in text with a high degree of accuracy.
* It is very fast. This makes it ideal for real-time applications, such as sentiment analysis of social media posts.
* It can handle multiple domains and tasks with a single model¹.
* It can generate high-quality texts that are relevant and informative⁴.
* It can adapt to different styles and tones of natural language⁴.
* It can perform sentiment analysis on texts and classify them as positive, negative, or neutral³.

Recent updates to OpenAI API usage costs including the introduction of function calling feature has made usage of Instruction tuned OpenAI models using Few Shot prompting quite economical than having an inhouse LLM model.



## **Demerits**

* It is expensive to train and use. This is because it requires a lot of computing power and data.
* It is less powerful and accurate than Text Davinci, which is the most capable model in the GPT-3 family¹.
* It may generate texts that are biased, inaccurate, or inappropriate due to the limitations of its training data or its inherent randomness⁴.
* It may not be able to handle complex reasoning or logic that requires external knowledge or common sense⁴.
* It can be biased. This is because it was trained on a dataset that was created by humans, who are all biased in their own way.
* It can be inaccurate. This is because it is still under development and is not perfect.

## **Opportunities for improvement**

* Making it more affordable. This could be done by making it more efficient or by finding ways to share the cost of training and use.
* Reducing its bias. This could be done by using a more diverse dataset or by developing better algorithms for detecting and removing bias.
* Improving its accuracy. This could be done by training it on a larger and more diverse dataset or by developing better algorithms for sentiment classification.
* Fine-tuning it on specific domains or tasks that require more specialized knowledge or skills¹.
* Incorporating more diverse and reliable sources of text data to reduce bias and increase accuracy⁴.
* Enhancing its ability to perform multimodal tasks that involve images, audio, or video inputs or outputs¹.
* Developing methods to evaluate and monitor its performance and quality on different natural language tasks⁴.

Overall, the text Curie model is a powerful tool for sentiment classification. It is accurate, fast, and efficient. However, it is still under development, and there are some areas where it could be improved.

Sure, I can try to write a few notes on instruction tuning and few-shot prompting. Here is one possible version:

## **Instruction Tuning and Few-Shot Prompting: Methods for Improving Zero-Shot and Few-Shot Learning in Language Models**

### Introduction

- Language models (LMs) are powerful tools that can perform various natural language processing (NLP) tasks without requiring task-specific training or adaptation.

- However, LMs still face challenges in generalizing to new tasks and datasets that they have not seen before, especially when the tasks are complex or require reasoning skills.

- Two methods that can help improve the zero-shot and few-shot learning abilities of LMs are instruction tuning and few-shot prompting.

- Instruction tuning is a method that fine-tunes a LM on a collection of datasets described via natural language instructions, and then evaluates it on unseen tasks using the same instructions.

- Few-shot prompting is a technique that uses demonstrations or examples in the prompt to guide the LM to better performance on a task.

- In this note, we will explain these methods in more detail and provide some examples of how they work.

### Instruction Tuning

- Instruction tuning is an emergent paradigm in NLP wherein natural language instructions are leveraged with LMs to induce zero-shot performance on unseen tasks.

- Instructions are natural language descriptions of the task or the desired output that the LM should produce. For example, if the task is to classify the sentiment of a text, the instruction could be "Tell me if this text is positive or negative".

- Instruction tuning involves fine-tuning a LM on a mixture of tasks phrased as instructions. The LM learns to perform these tasks by seeing examples of inputs and outputs during fine-tuning. For example, the LM could see texts and their sentiment labels as inputs and outputs for the instruction above.

- At inference time, the LM is evaluated on unseen tasks using the same instructions. The LM tries to generate the correct output for the given input based on the instruction. For example, the LM could see a new text and try to classify its sentiment using the same instruction as before.

- Instruction tuning can substantially improve zero-shot performance on unseen tasks and datasets in both large and small LMs. It can also outperform zero-shot and few-shot GPT-3 on some tasks.

### Few-Shot Prompting

- Few-shot prompting is a technique that uses demonstrations or examples in the prompt to enable in-context learning where we provide demonstrations in the prompt to steer the LM to better performance on a task.

- Demonstrations are pairs of inputs and outputs that show how the task should be performed. They serve as conditioning for subsequent examples where the LM is expected to generate a response. For example, if the task is to use a new word in a sentence, the demonstration could be a pair of a word and a sentence that uses it.

- Few-shot prompting involves providing one or more demonstrations in the prompt along with an input for which we want the LM to generate an output. The LM learns from the demonstrations and tries to generate a similar output for the input. For example, the prompt could provide an example of a sentence that uses a word, and then ask the LM to do the same for another word.

- Few-shot prompting can improve zero-shot performance on many tasks, especially when LMs are scaled to a sufficient size. It can also provide additional information and perspectives on the task that may not be captured by instructions alone.

### Conclusion

- Instruction tuning and few-shot prompting are both ways of leveraging natural language instructions with LMs to achieve better zero-shot or few-shot learning.

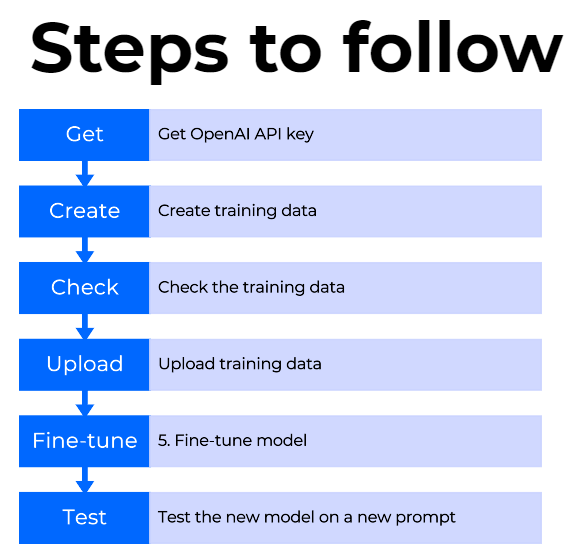
- They can be combined or used separately depending on the task and the data available.

- They can help LMs generalize to new tasks and datasets without requiring any additional training or prompt engineering.

## **Fine tuning open-ai models with own data.**

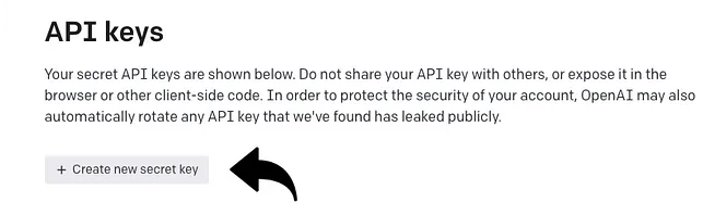
-Using Few Shot prompting

* Fine-tuning LLMs: Fine-tuning language models (LLMs) using OpenAI is a powerful technique that enables customization and adaptation of pre-trained models to specific tasks or domains.
* OpenAI's Fine-Tuning Capability: OpenAI provides the infrastructure and tools necessary to fine-tune their pre-trained models, such as GPT-3, allowing developers to leverage the benefits of these state-of-the-art models while tailoring them to their specific needs.
* Task-Specific Adaptation: Fine-tuning LLMs involves training the pre-trained models on task-specific datasets, enabling the model to learn domain-specific patterns, terminologies, and behaviors, resulting in more contextually relevant and accurate outputs.
* Diverse Range of Applications: Fine-tuning LLMs with OpenAI has proven effective across various natural language processing (NLP) applications, including text generation, sentiment analysis, machine translation, question answering, summarization, and more,



### Get Open API Key**​**

* Visit <https://platform.openai.com/,>​
* log in and click on your avatar and **View API keys.**​
* Then create a **new secret key** and save it for the request:​



* Now we have all the credentials needed to make an API request.​  
  ​​

### 2. Create training data​

* The next step is to create training data to teach GPT-3 what you’d like to say. The data need to be a JSONL document with a new prompt and the ideal generated text:​

{"prompt": "<question>", "completion": "<ideal answer>"}​  
{"prompt": "<question>", "completion": "<ideal answer>"}​  
{"prompt": "<question>", "completion": "<ideal answer>"}​

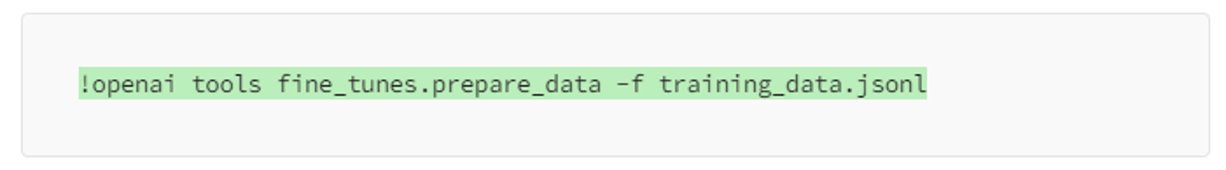
* It is recommended to have at least a few hundred samples for training, depending on the degree of variability in the data.​
* Please note that for sentiment classification,**no preprocessing and feature engineering needs to be performed**in the data as curie is already trained on large amounts of text data and can capture semantic similarity and information retrieval well.​

### 3. Python implementation

i. import the required libraries



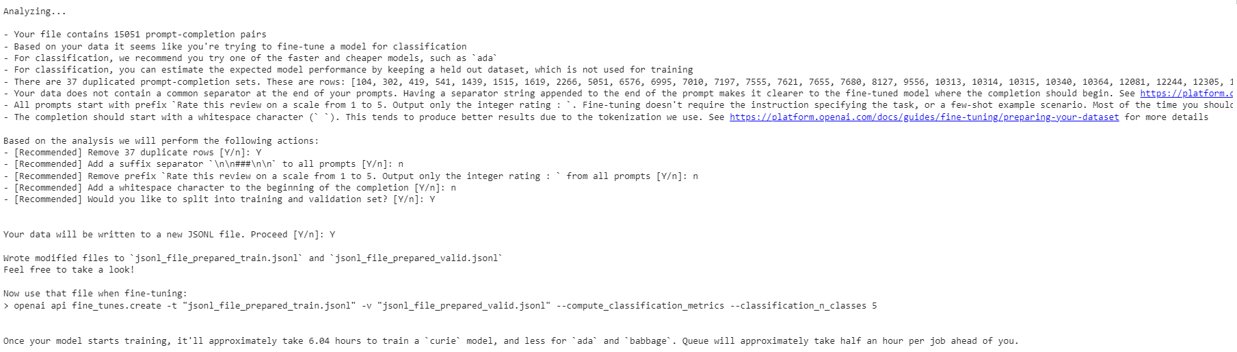
* We can check the training data using a CLI data preparation tool provided by OpenAI. It gives us suggestions about how one can reformat the training data.​
* Run the below line in the Jupyter notebook:



* Post running the CLI checks we would receive suggestions like the ones mentioned in the picture below: ​

[ Please note that this step need not be executed if the jsonl format is appropriately maintained in the training and test data.] ​

[We'll also see the approximate time depending on how many jobs are ahead of us in the queue]​



​

### 4. Upload training data​

****

* If we check the response, we can see the file id which we'll need in the next step when we're training the model:​

****

Use this file id in the next step, where we'll fine-tune a model.​

### 5. Fine-tune model​

​

​

​

Start the fine-tuning by running this command:​

****

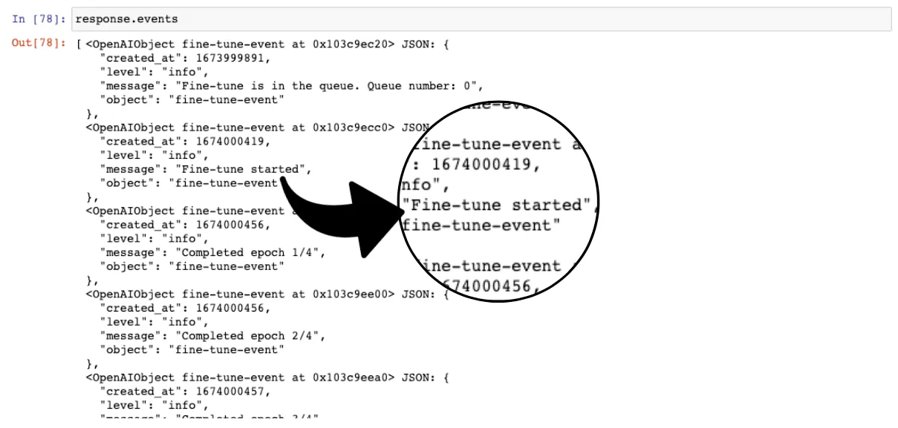
The default model is Curie. But if you'd like to use DaVinci instead, then add it as a base model to fine-tune like this:​

****

The first response will look like below :​

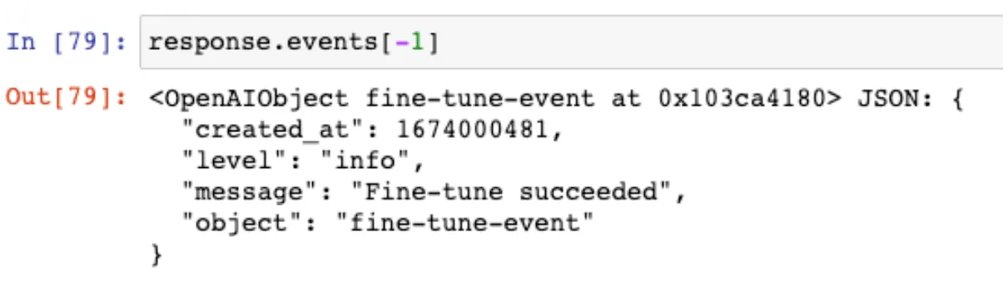
****

* If we run response.events, we'll periodically see the current queue number during the process.​
* When our fine-tuning job is first in line, the fine-tuning event starts:​

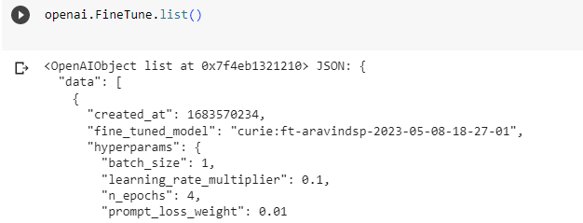
****

* check the fine-tuning events by running the following snippet:​

****

****

* Once the fine-tuning is finished, save the name of the fine-tuned model into a variable:​

****

****

# **Suggestions to Customers based on Data Analysis**

## **Atlanta Airport**

* Expanding security checkpoints to reduce wait times.
* Adding more food and drink options, and comfortable seating.
* Investing in new baggage handling technology to improve efficiency.
* Training staff on customer service.
* Investing in new technology to improve traffic flow and reduce delays.

# **REFERENCES**

(1) Azure OpenAI Service models - Azure OpenAI | Microsoft Learn. https://learn.microsoft.com/en-us/azure/cognitive-services/openai/concepts/models.

(2) OpenAI Platform. https://platform.openai.com/docs/models.

(3) A Guide to Text Classification and Sentiment Analysis. https://towardsdatascience.com/a-guide-to-text-classification-and-sentiment-analysis-2ab021796317.

(4) Sentiment Classification of News Text Data Using Intelligent Model. https://www.frontiersin.org/articles/10.3389/fpsyg.2021.758967/full.

(5) Applied Sciences | Free Full-Text | Sentiment Classification Using .... <https://www.mdpi.com/2076-3417/9/11/2347>.

[6] F L MODELS ARE ZERO-SHOT L - arXiv.org. https://arxiv.org/pdf/2109.01652.pdf.

[7] Few-Shot Prompting | Prompt Engineering Guide. https://www.promptingguide.ai/techniques/fewshot.

[8] Improving Zero and Few-shot Generalization in Dialogue through .... https://www.researchgate.net/publication/360859188\_Improving\_Zero\_and\_Few-shot\_Generalization\_in\_Dialogue\_through\_Instruction\_Tuning.

[9] Few-shot learning in practice: GPT-Neo and the 🤗 Accelerated Inference API. <https://huggingface.co/blog/few-shot-learning-gpt-neo-and-inference-api>.

[10] Wei et al. (2021). Finetuned Language Models Are Zero-Shot Learners. https://arxiv.org/abs/2109.01652

[11] Brown et al. (2020). Language Models are Few-Shot Learners. https://arxiv.org/abs/2005.14165

[12] Min et al. (2021). Improving Zero and Few-Shot Generalization in Dialogue through Instruction Tuning. https://arxiv.org/abs/2109.01652

