**Aprendizaje fino**

Learn how to customize a model for your application.

Aprenda cómo personalizar un modelo para su aplicación.

**Introducción**

Fine-tuning lets you get more out of the models available through the API by providing:

El aprendizaje fino (fine-tuning) le permite obtener más de los modelos disponibles a través de la API al proporcionar:

1. Higher quality results than prompt design
2. Ability to train on more examples than can fit in a prompt
3. Token savings due to shorter prompts
4. Lower latency requests
5. Resultados de mayor calidad que el diseño de la indicación
6. Capacidad para entrenar con más ejemplos que los que caben en una indicación
7. Ahorro de tokens debido a indicaciones más cortas
8. Solicitudes de latencia más baja

GPT-3 has been pre-trained on a vast amount of text from the open internet. When given a prompt with just a few examples, it can often intuit what task you are trying to perform and generate a plausible completion. This is often called "few-shot learning."

GPT-3 ha sido pre-entrenado en una gran cantidad de texto de Internet. Cuando se le proporciona una indicación con solo unos pocos ejemplos, a menudo puede intuir qué tarea está intentando realizar y generar una finalización plausible. Esto a menudo se llama "aprendizaje de pocas muestras".

Fine-tuning improves on few-shot learning by training on many more examples than can fit in the prompt, letting you achieve better results on a wide number of tasks. **Once a model has been fine-tuned, you won't need to provide examples in the prompt anymore.** This saves costs and enables lower-latency requests.

El aprendizaje fino mejora el aprendizaje de pocas muestras al entrenar con muchos más ejemplos de los que caben en la indicación, lo que le permite obtener mejores resultados en una amplia cantidad de tareas. **Una vez que se haya realizado el aprendizaje fino en un modelo, ya no será necesario proporcionar ejemplos en la indicación.** Esto ahorra costos y permite solicitudes con una latencia más baja.

At a high level, fine-tuning involves the following steps:

A nivel general, el proceso de fine-tuning implica los siguientes pasos:

1. Prepare and upload training data
2. Train a new fine-tuned model
3. Use your fine-tuned model
4. Preparar y subir los datos de entrenamiento
5. Entrenar un nuevo modelo fine-tuned
6. Utilizar el modelo fine-tuned

Visit our [pricing page](https://openai.com/api/pricing) to learn more about how fine-tuned model training and usage are billed.

Visita nuestra página de precios para obtener más información sobre cómo se facturan el entrenamiento y uso de modelos fine-tuned.

**[What models can be fine-tuned?](https://platform.openai.com/docs/guides/fine-tuning/what-models-can-be-fine-tuned)**

**¿Qué modelos se pueden ajustar?**

Fine-tuning is currently only available for the following base models: davinci, curie, babbage, and ada. These are the original models that do not have any instruction following training (like text-davinci-003 does for example). You are also able to [continue fine-tuning a fine-tuned model](https://platform.openai.com/docs/guides/fine-tuning/continue-fine-tuning-from-a-fine-tuned-model) to add additional data without having to start from scratch.

Actualmente, el ajuste fino solo está disponible para los siguientes modelos base: davinci, curie, babbage y ada. Estos son los modelos originales que no tienen ninguna instrucción después del entrenamiento (como lo hace, por ejemplo, text-davinci-003). También puede continuar ajustando un modelo fino ajustado para agregar datos adicionales sin tener que comenzar desde cero.

**Instalación**

We recommend using our OpenAI command-line interface (CLI). To install this, run

Recomendamos usar nuestra interfaz de línea de comandos (CLI) de OpenAI. Para instalar esto, ejecute

pip install --upgrade openai

(The following instructions work for version 0.9.4 and up. Additionally, the OpenAI CLI requires python 3.)

(Las siguientes instrucciones funcionan para la versión 0.9.4 en adelante. Además, la CLI de OpenAI requiere python 3.)

Set your OPENAI\_API\_KEY environment variable by adding the following line into your shell initialization script (e.g. .bashrc, zshrc, etc.) or running it in the command line before the fine-tuning command:

Configure su variable de entorno OPENAI\_API\_KEY agregando la siguiente línea en su script de inicialización de shell (por ejemplo, .bashrc, zshrc, etc.) o ejecutándolo en la línea de comando antes del comando de ajuste fino:

export OPENAI\_API\_KEY = "<OPENAI\_API\_KEY>"

**[Prepare training data](https://platform.openai.com/docs/guides/fine-tuning/prepare-training-data)**

**Preparar los datos de entrenamiento**

Training data is how you teach GPT-3 what you'd like it to say.

Los datos de entrenamiento son la forma en que enseña a GPT-3 lo que le gustaría decir.

Your data must be a [JSONL](https://jsonlines.org/) document, where each line is a prompt-completion pair corresponding to a training example. You can use our [CLI data preparation tool](https://platform.openai.com/docs/guides/fine-tuning/cli-data-preparation-tool) to easily convert your data into this file format.

Sus datos deben ser un documento JSONL, donde cada línea es un par de prompts y completions correspondiente a un ejemplo de entrenamiento. Puede usar nuestra herramienta de preparación de datos CLI para convertir fácilmente sus datos a este formato de archivo.

1{"prompt": "<prompt text>", "completion": "<ideal generated text>"}

2{"prompt": "<prompt text>", "completion": "<ideal generated text>"}

3{"prompt": "<prompt text>", "completion": "<ideal generated text>"}

4...

Designing your prompts and completions for fine-tuning is different from designing your prompts for use with our base models (Davinci, Curie, Babbage, Ada). In particular, while prompts for base models often consist of multiple examples ("few-shot learning"), for fine-tuning, each training example generally consists of a single input example and its associated output, without the need to give detailed instructions or include multiple examples in the same prompt.

Diseñar sus prompts y completions para el ajuste fino es diferente de diseñar sus prompts para usar con nuestros modelos base (Davinci, Curie, Babbage, Ada). En particular, mientras que los prompts para los modelos base a menudo consisten en múltiples ejemplos ("aprendizaje con pocos ejemplos"), para el ajuste fino, cada ejemplo de entrenamiento generalmente consiste en un solo ejemplo de entrada y su salida asociada, sin necesidad de dar instrucciones detalladas o incluir múltiples ejemplos en el mismo prompt.

For more detailed guidance on how to prepare training data for various tasks, please refer to our [preparing your dataset](https://platform.openai.com/docs/guides/fine-tuning/preparing-your-dataset) best practices.

Para obtener una guía detallada sobre cómo preparar datos de entrenamiento para varias tareas, consulte nuestras mejores prácticas para preparar su conjunto de datos.

The more training examples you have, the better. We recommend having at least a couple hundred examples. In general, we've found that each doubling of the dataset size leads to a linear increase in model quality.

Cuantos más ejemplos de entrenamiento tenga, mejor. Recomendamos tener al menos un par de cientos de ejemplos. En general, hemos encontrado que cada duplicación del tamaño del conjunto de datos conduce a un aumento lineal en la calidad del modelo.

**[CLI data preparation tool](https://platform.openai.com/docs/guides/fine-tuning/cli-data-preparation-tool)**

We developed a tool which validates, gives suggestions and reformats your data:

Hemos desarrollado una herramienta que valida, sugiere cambios y reformatea sus datos:

openai tools fine\_tunes.prepare\_data -f <LOCAL\_FILE>

This tool accepts different formats, with the only requirement that they contain a prompt and a completion column/key. You can pass a **CSV, TSV, XLSX, JSON** or **JSONL** file, and it will save the output into a JSONL file ready for fine-tuning, after guiding you through the process of suggested changes.

Esta herramienta acepta diferentes formatos, con el único requisito de que contengan una columna o clave de entrada y una de salida. Puede pasar un archivo CSV, TSV, XLSX, JSON o JSONL, y guardará la salida en un archivo JSONL listo para el ajuste fino, después de guiarlo a través del proceso de cambios sugeridos.

**[Create a fine-tuned model](https://platform.openai.com/docs/guides/fine-tuning/create-a-fine-tuned-model)**

**Crear un modelo afinado**

The following assumes you've already prepared training data following the [above instructions](https://platform.openai.com/docs/guides/fine-tuning/prepare-training-data).

Lo siguiente asume que ya ha preparado los datos de entrenamiento siguiendo las instrucciones anteriores.

Start your fine-tuning job using the OpenAI CLI:

Inicie su trabajo de ajuste fino utilizando la CLI de OpenAI:

openai api fine\_tunes.create -t <TRAIN\_FILE\_ID\_OR\_PATH> -m <BASE\_MODEL>

Where BASE\_MODEL is the name of the base model you're starting from (ada, babbage, curie, or davinci). You can customize your fine-tuned model's name using the [suffix parameter](https://platform.openai.com/docs/guides/fine-tuning/customize-your-model-name).

Donde BASE\_MODEL es el nombre del modelo base desde el que está comenzando (ada, babbage, curie o davinci). Puede personalizar el nombre de su modelo afinado agregando un sufijo.

Running the above command does several things:

La ejecución del comando anterior realiza varias acciones:

1. Uploads the file using the [files API](https://platform.openai.com/docs/api-reference/files) (or uses an already-uploaded file)
2. Creates a fine-tune job
3. Streams events until the job is done (this often takes minutes, but can take hours if there are many jobs in the queue or your dataset is large)
4. Carga el archivo utilizando la API de archivos (o usa un archivo ya cargado).
5. Crea un trabajo de ajuste fino.
6. Transmite eventos hasta que se complete el trabajo (esto a menudo lleva minutos, pero puede tardar horas si hay muchos trabajos en la cola o si su conjunto de datos es grande).

Every fine-tuning job starts from a base model, which defaults to curie. The choice of model influences both the performance of the model and the cost of running your fine-tuned model. Your model can be one of: ada, babbage, curie, or davinci. Visit our [pricing page](https://openai.com/api/pricing/#faq-fine-tuning-pricing-calculation) for details on fine-tune rates.

Cada trabajo de ajuste fino comienza desde un modelo base, que por defecto es curie. La elección del modelo influye tanto en el rendimiento del modelo como en el costo de ejecutar su modelo afinado. Su modelo puede ser uno de: ada, babbage, curie o davinci. Visite nuestra página de precios para obtener detalles sobre las tarifas de ajust.

After you've started a fine-tune job, it may take some time to complete. Your job may be queued behind other jobs on our system, and training our model can take minutes or hours depending on the model and dataset size. If the event stream is interrupted for any reason, you can resume it by running:

Después de haber iniciado un trabajo de ajuste fino, puede tomar algún tiempo completarlo. Es posible que su trabajo esté en cola detrás de otros trabajos en nuestro sistema y que el entrenamiento de nuestro modelo lleve minutos u horas dependiendo del modelo y del tamaño del conjunto de datos. Si el flujo de eventos se interrumpe por alguna razón, puede reanudarlo ejecutando:

openai api fine\_tunes.follow -i <YOUR\_FINE\_TUNE\_JOB\_ID>

When the job is done, it should display the name of the fine-tuned model.

Cuando el trabajo esté completo, debería mostrar el nombre del modelo afinado.

In addition to creating a fine-tune job, you can also list existing jobs, retrieve the status of a job, or cancel a job.

Además de crear un trabajo de ajuste fino, también puede enumerar los trabajos existentes, recuperar el estado de un trabajo o cancelar un trabajo.

1 # List all created fine-tunes

2 openai api fine\_tunes.list

3 # Retrieve the state of a fine-tune. The resulting object includes

4 # job status (which can be one of pending, running, succeeded, or failed)

5 # and other information

6 openai api fine\_tunes.get -i <YOUR\_FINE\_TUNE\_JOB\_ID>

7 # Cancel a job

8 openai api fine\_tunes.cancel -i <YOUR\_FINE\_TUNE\_JOB\_ID>

9 # List all created fine-tunes

10 openai api fine\_tunes.list

1 # Enumerar todos los ajustes finos creados

2 openai api fine\_tunes.list

3 # Recuperar el estado de un ajuste fino. El objeto resultante

4 # incluye estado del trabajo (que puede ser pendiente, ejecutándose, exitoso o fallido)

5 # y otra información

6 openai api fine\_tunes.get -i <YOUR\_FINE\_TUNE\_JOB\_ID>

7 # Cancelar un trabajo

8 openai api fine\_tunes.cancel -i <YOUR\_FINE\_TUNE\_JOB\_ID>

9 # Listar todas las afinaciones finas creadas

10 openai api fine\_tunes.list

**[Use a fine-tuned model](https://platform.openai.com/docs/guides/fine-tuning/use-a-fine-tuned-model)**

**Usar un modelo afinado**

When a job has succeeded, the fine\_tuned\_model field will be populated with the name of the model. You may now specify this model as a parameter to our [Completions API](https://platform.openai.com/docs/api-reference/completions), and make requests to it using the [Playground](https://platform.openai.com/playground).

Cuando un trabajo ha sido exitoso, el campo fine\_tuned\_model se completará con el nombre del modelo. Ahora puede especificar este modelo como un parámetro de nuestra API de Completions y realizar solicitudes a través del Playground.

After your job first completes, it may take several minutes for your model to become ready to handle requests. If completion requests to your model time out, it is likely because your model is still being loaded. If this happens, try again in a few minutes.

Después de que su trabajo se haya completado por primera vez, puede tomar varios minutos para que su modelo esté listo para manejar solicitudes. Si las solicitudes de completitud a su modelo se agotan, es probable que su modelo aún se esté cargando. Si esto sucede, inténtelo de nuevo en unos minutos.

You can start making requests by passing the model name as the model parameter of a completion request:

Puede comenzar a realizar solicitudes pasando el nombre del modelo como parámetro del modelo de una solicitud de completitud:

OpenAI CLI:

openai api completions.create -m <FINE\_TUNED\_MODEL> -p <YOUR\_PROMPT>

cURL:

1 curl https://api.openai.com/v1/completions \

2 -H "Authorization: Bearer $OPENAI\_API\_KEY" \

3 -H "Content-Type: application/json" \

4 -d '{"prompt": YOUR\_PROMPT, "model": FINE\_TUNED\_MODEL}'

Python:

1 import openai

2 openai.Completion.create(

3 model=FINE\_TUNED\_MODEL,

4 prompt=YOUR\_PROMPT)

You may continue to use all the other [Completions](https://platform.openai.com/docs/api-reference/completions) parameters like temperature, frequency\_penalty, presence\_penalty, etc, on these requests to fine-tuned models.

Puedes seguir utilizando todos los otros parámetros de Completions, como temperature, frequency\_penalty, presence\_penalty, etc., en estas solicitudes a modelos afinados.

**[Delete a fine-tuned model](https://platform.openai.com/docs/guides/fine-tuning/delete-a-fine-tuned-model)**

**Eliminar un modelo** **afinado**

To delete a fine-tuned model, you must be designated an "owner" within your organization.

Para eliminar un modelo afinado, debes ser designado como "propietario" dentro de tu organización.

OpenAI CLI:

openai api models.delete -i <FINE\_TUNED\_MODEL>

cURL:

curl -X "DELETE" https://api.openai.com/v1/models/<FINE\_TUNED\_MODEL> \

-H "Authorization: Bearer $OPENAI\_API\_KEY"

Python:

import openai

openai.Model.delete(FINE\_TUNED\_MODEL)

**[Preparing your dataset](https://platform.openai.com/docs/guides/fine-tuning/preparing-your-dataset)**

**Preparando su conjunto de datos**

Fine-tuning is a powerful technique to create a new model that's specific to your use case. Before fine-tuning your model, we strongly recommend reading these best practices and [specific guidelines](https://platform.openai.com/docs/guides/fine-tuning/specific-guidelines) for your use case below.

El ajuste fino es una técnica poderosa para crear un nuevo modelo específico para su caso de uso. Antes de ajustar fino su modelo, recomendamos encarecidamente leer estas mejores prácticas y pautas específicas para su caso de uso a continuación.

**[Data formatting](https://platform.openai.com/docs/guides/fine-tuning/data-formatting)**

**Formato de datos**

To fine-tune a model, you'll need a set of training examples that each consist of a single input ("prompt") and its associated output ("completion"). This is notably different from using our base models, where you might input detailed instructions or multiple examples in a single prompt.

Para ajustar fino un modelo, necesitará un conjunto de ejemplos de entrenamiento que consistan en una única entrada ("indicación") y su salida asociada ("completado"). Esto es notablemente diferente al uso de nuestros modelos base, donde podría ingresar instrucciones detalladas o múltiples ejemplos en una sola indicación.

* Each prompt should end with a fixed separator to inform the model when the prompt ends and the completion begins. A simple separator which generally works well is \n\n###\n\n. The separator should not appear elsewhere in any prompt.
* Cada indicación debe finalizar con un separador fijo para informar al modelo cuándo finaliza la indicación y comienza el completado. Un separador simple que generalmente funciona bien es \n\n###\n\n. El separador no debe aparecer en ninguna otra indicación.
* Each completion should start with a whitespace due to our [tokenization](https://platform.openai.com/tokenizer), which tokenizes most words with a preceding whitespace.
* Cada completado debe comenzar con un espacio en blanco debido a nuestra tokenización, que tokeniza la mayoría de las palabras con un espacio en blanco precedente.
* Each completion should end with a fixed stop sequence to inform the model when the completion ends. A stop sequence could be \n, ###, or any other token that does not appear in any completion.
* Cada completado debe finalizar con una secuencia de detención fija para informar al modelo cuándo finaliza el completado. Una secuencia de detención podría ser \n, ###, o cualquier otro token que no aparezca en ningún completado.
* For inference, you should format your prompts in the same way as you did when creating the training dataset, including the same separator. Also specify the same stop sequence to properly truncate the completion.
* Para la inferencia, debe formatear sus indicaciones de la misma manera que lo hizo al crear el conjunto de datos de entrenamiento, incluyendo el mismo separador. También especifique la misma secuencia de detención para truncar adecuadamente el completado.

**[General best practices](https://platform.openai.com/docs/guides/fine-tuning/general-best-practices)**

**Mejores prácticas generales**

Fine-tuning performs better with more high-quality examples. To fine-tune a model that performs better than using a high-quality prompt with our base models, you should provide at least a few hundred high-quality examples, ideally vetted by human experts. From there, performance tends to linearly increase with every doubling of the number of examples. Increasing the number of examples is usually the best and most reliable way of improving performance.

El ajuste fino funciona mejor con más ejemplos de alta calidad. Para ajustar fino un modelo que funcione mejor que usar una indicación de alta calidad con nuestros modelos base, debe proporcionar al menos unos cientos de ejemplos de alta calidad, idealmente evaluados por expertos humanos. A partir de ahí, el rendimiento tiende a aumentar linealmente con cada duplicación del número de ejemplos. Aumentar el número de ejemplos suele ser la mejor y más confiable manera de mejorar el rendimiento.

Classifiers are the easiest models to get started with. For classification problems we suggest using ada, which generally tends to perform only very slightly worse than more capable models once fine-tuned, whilst being significantly faster and cheaper.

Los clasificadores son los modelos más fáciles para comenzar. Para problemas de clasificación, sugerimos usar ada, que generalmente tiende a funcionar solo ligeramente peor que los modelos más capaces una vez ajustados finamente, mientras que es significativamente más rápido y más económico.

If you are fine-tuning on a pre-existing dataset rather than writing prompts from scratch, be sure to manually review your data for offensive or inaccurate content if possible, or review as many random samples of the dataset as possible if it is large.

Si está ajustando fino en un conjunto de datos preexistente en lugar de escribir indicaciones desde cero, asegúrese de revisar manualmente sus datos en busca de contenido ofensivo o inexacto si es posible, o revise tantas muestras aleatorias del conjunto de datos como sea posible si es grande.

**[Specific guidelines](https://platform.openai.com/docs/guides/fine-tuning/specific-guidelines)**

**Pautas específicas**

Fine-tuning can solve a variety of problems, and the optimal way to use it may depend on your specific use case. Below, we've listed the most common use cases for fine-tuning and corresponding guidelines.

El ajuste fino puede resolver una variedad de problemas, y la forma óptima de usarlo puede depender de su caso de uso específico. A continuación, hemos enumerado los casos de uso más comunes para el ajuste fino y las pautas correspondientes.

* + [Classification](https://platform.openai.com/docs/guides/fine-tuning/classification)
  + [Is the model making untrue statements?](https://platform.openai.com/docs/guides/fine-tuning/case-study-is-the-model-making-untrue-statements)
  + [Sentiment analysis](https://platform.openai.com/docs/guides/fine-tuning/case-study-sentiment-analysis)
  + [Categorization for email triage](https://platform.openai.com/docs/guides/fine-tuning/case-study-categorization-for-email-triage)
  + [Conditional generation](https://platform.openai.com/docs/guides/fine-tuning/conditional-generation)
  + [Write an engaging ad based on a Wikipedia article](https://platform.openai.com/docs/guides/fine-tuning/case-study-write-an-engaging-ad-based-on-a-wikipedia-article)
  + [Entity extraction](https://platform.openai.com/docs/guides/fine-tuning/case-study-entity-extraction)
  + [Customer support chatbot](https://platform.openai.com/docs/guides/fine-tuning/case-study-customer-support-chatbot)
  + [Product description based on a technical list of properties](https://platform.openai.com/docs/guides/fine-tuning/case-study-product-description-based-on-a-technical-list-of-properties)
  + Clasificación
  + ¿El modelo está haciendo declaraciones falsas?
  + Análisis de sentimientos
  + Categorización para la organización de correo electrónico
  + Generación condicional
  + Escribir un anuncio atractivo basado en un artículo de Wikipedia
  + Extracción de entidades
  + Chatbot de soporte al cliente
  + Descripción de productos basada en una lista técnica de propiedades

**[Classification](https://platform.openai.com/docs/guides/fine-tuning/classification)**

**Clasificación**

In classification problems, each input in the prompt should be classified into one of the predefined classes. For this type of problem, we recommend:

En problemas de clasificación, cada entrada en la solicitud debe ser clasificada en una de las clases predefinidas. Para este tipo de problema, recomendamos:

* + Use a separator at the end of the prompt, e.g. \n\n###\n\n. Remember to also append this separator when you eventually make requests to your model.
  + Usar un separador al final de la solicitud, por ejemplo, \n\n###\n\n. Recuerda agregar este separador también cuando hagas solicitudes a tu modelo.
  + Choose classes that map to a single [token](https://platform.openai.com/tokenizer). At inference time, specify max\_tokens=1 since you only need the first token for classification.
  + Elegir clases que se correspondan con un solo token. En el momento de la inferencia, especifica max\_tokens=1 ya que solo necesitas el primer token para la clasificación.
  + Ensure that the prompt + completion doesn't exceed 2048 tokens, including the separator
  + Asegurarse de que la solicitud + respuesta no supere los 2048 tokens, incluyendo el separador.
  + Aim for at least ~100 examples per class
  + Apuntar a al menos ~100 ejemplos por clase.
  + To get class log probabilities you can specify logprobs=5 (for 5 classes) when using your model
  + Para obtener las probabilidades de registro de la clase, puedes especificar logprobs=5 (para 5 clases) al usar tu modelo.
  + Ensure that the dataset used for finetuning is very similar in structure and type of task as what the model will be used for
  + Asegurarse de que el conjunto de datos utilizado para el ajuste fino sea muy similar en estructura y tipo de tarea a lo que se utilizará el modelo.

**Case study: Is the model making untrue statements?**

**Caso de Estudio: ¿El modelo está haciendo afirmaciones falsas?**

Let's say you'd like to ensure that the text of the ads on your website mention the correct product and company. In other words, you want to ensure the model isn't making things up. You may want to fine-tune a classifier which filters out incorrect ads. The dataset might look something like the following:

Supongamos que deseas asegurarte de que el texto de los anuncios en tu sitio web mencione el producto y la empresa correctos. En otras palabras, quieres asegurarte de que el modelo no invente cosas. Puedes querer afinar un clasificador que filtre los anuncios incorrectos.

The dataset might look something like the following:

El conjunto de datos podría tener un aspecto similar al siguiente:

{"prompt":"Company: BHFF insurance\nProduct: allround insurance\nAd:One stop shop for all your insurance needs!\nSupported:", "completion":" yes"}

{"prompt":"Company: Loft conversion specialists\nProduct: -\nAd:Straight teeth in weeks!\nSupported:", "completion":" no"}

In the example above, we used a structured input containing the name of the company, the product, and the associated ad. As a separator we used \nSupported: which clearly separated the prompt from the completion. With a sufficient number of examples, the separator doesn't make much of a difference (usually less than 0.4%) as long as it doesn't appear within the prompt or the completion.

En el ejemplo anterior, utilizamos una entrada estructurada que contiene el nombre de la compañía, el producto y el anuncio asociado. Como separador, utilizamos \nSupported:, que separa claramente la entrada de la salida. Con un número suficiente de ejemplos, el separador no hace mucha diferencia (generalmente menos del 0,4%), siempre y cuando no aparezca dentro de la entrada o la salida.

For this use case we fine-tuned an ada model since it will be faster and cheaper, and the performance will be comparable to larger models because it is a classification task.

Para este caso de uso, afinamos un modelo ada ya que será más rápido y económico, y el rendimiento será comparable a modelos más grandes porque es una tarea de clasificación.

Now we can query our model by making a Completion request.

Ahora podemos consultar nuestro modelo haciendo una solicitud de Completado.

1 curl https://api.openai.com/v1/completions \

2 -H 'Content-Type: application/json' \

3 -H 'Authorization: Bearer YOUR\_API\_KEY' \

4 -d '{

5 "prompt": "Company: Reliable accountants Ltd\nProduct: Personal Tax help\nAd:Best 6 advice in town!\nSupported:",

7 "max\_tokens": 1,

8 "model": "YOUR\_FINE\_TUNED\_MODEL\_NAME"

9 }'

Which will return either yes or no.

Lo cual devolverá sí o no.

**[Case study: Sentiment analysis](https://platform.openai.com/docs/guides/fine-tuning/case-study-sentiment-analysis)**

**Caso de estudio: Análisis** **de sentimiento**

Let's say you'd like to get a degree to which a particular tweet is positive or negative. The dataset might look something like the following:

Imaginemos que queremos determinar el grado de positividad o negatividad de un tweet en particular. El conjunto de datos podría tener un aspecto similar al siguiente:

1 {"prompt":"Overjoyed with the new iPhone! ->", "completion":" positive"}

2 {"prompt":"@lakers disappoint for a third straight night https://t.co/38EFe43 ->", "completion":" negative"}

Once the model is fine-tuned, you can get back the log probabilities for the first completion token by setting logprobs=2 on the completion request. The higher the probability for positive class, the higher the relative sentiment.

Una vez que el modelo está afinado, puedes obtener las probabilidades logarítmicas para el primer token de completado estableciendo logprobs=2 en la solicitud de completado. Cuanto mayor sea la probabilidad para la clase positiva, mayor será el sentimiento relativo positivo.

Now we can query our model by making a Completion request.

Ahora podemos consultar nuestro modelo haciendo una solicitud de completado.

1 curl https://api.openai.com/v1/completions \

2 -H 'Content-Type: application/json' \

3 -H 'Authorization: Bearer YOUR\_API\_KEY' \

4 -d '{

5 "prompt": "https://t.co/f93xEd2 Excited to share my latest blog post! ->",

6 "max\_tokens": 1,

7 "model": "YOUR\_FINE\_TUNED\_MODEL\_NAME"

8 }'

LUCI DATE (https://www.youtube.com/watch?v=uFiI5fK-7B4)

En este video discutiremos un par de métodos fáciles para crear prompts y completions, así como algunos ejemplos simples de manipulación de texto para que puedas construir pipelines automatizados para el fine tuning.

Hay que señalar que el método más sencillo de crear prompts y completions es simplemente poblar manualmente una hoja de cálculo de Excel con dos columnas: una se llamará prompts y la otra completions.

No te recomiendo hacer esto, sino que mires algunas de las formas más automatizadas y escalables de escribir pipelines. Este enfoque funcionará y podrás ajustar un modelo de esta manera, pero el modelo final no se comportará de manera diferente si lo has entrenado manualmente o si has utilizado el reconocimiento de voz de los podcasts (más sobre eso en el próximo video).

Sin embargo, la actualización manual de una hoja de cálculo no escalará, y la razón por la que te interesa la IA en primer lugar es por los multiplicadores de eficiencia que obtienes, por lo que recomiendo que no lo hagas o que solo lo hagas como último recurso absoluto. Por cierto, lo harás y, si soy completamente honesto, yo también lo he hecho. Entonces, con esa admisión vergonzosa fuera del camino, veamos algo más sistemático.

Compartiré contigo algunas herramientas que puedes usar para manipular texto, así como herramientas para recuperar pasajes relevantes de texto de Internet. En nuestro último video discutimos un par de razones por las que querríamos encontrar contenido:

* + En primer lugar, nos gustaría que nuestro contenido fuera lo más actualizado posible para que sea relevante.
  + En segundo lugar, nos gustaría hacer que nuestro modelo de NLP sea menos generalista y más especializado en nuestro campo específico, como los mercados de capital.

En consecuencia, querremos entrenarlo en artículos recientes y relevantes mediante el ajuste fino en prompts y completions relevantes. Durante el entrenamiento, hacemos que el modelo actualice sus matrices de clave de consulta y valor en sus cabezas de atención. Estas matrices actualizadas Q, K y V significarán que, durante la inferencia, el modelo podrá generar una salida que sea más relevante y actualizada.

Por lo tanto, al ajustar fino nuestro Transformer con un prompt de un título de artículo y una finalización del texto del artículo, estamos actualizando las cabezas de atención en el Transformer. De manera similar, si usamos una oración individual como prompt y las cinco o diez oraciones que siguen como nuestra finalización, estamos proporcionando al modelo actualizado Q, K y V matrices que son más representativas de los desarrollos recientes en nuestro campo.

Cuando ajustamos finamente un Transformer, le enseñamos cosas nuevas y la manifestación de ese aprendizaje son matrices actualizadas de clave de consulta y valor.

Hay **dos problemas** específicos que debemos resolver:

* + En primer lugar, queremos poder obtener texto relevante de internet para entrenar y modelar nuestro modelo. Este texto probablemente estará en forma de artículos e informes que usaremos para construir nuestras prompts y nuestras completions. En lugar de tener estos artículos con un formato agradable en nuestro navegador, en cambio, nos gustaría que solo el texto de estos artículos aparezca en una aplicación que escribimos para que podamos generar algunos prompts y completions a partir de este texto.
  + En segundo lugar, una vez que tenemos este texto en nuestra aplicación, necesitamos hacer alguna manipulación en él para que podamos descomponerlo en prompts y completions. Por manipulaciones me refiero a cosas como arrancar títulos y encabezados de secciones que son buenos candidatos para prompts. De manera similar, los párrafos de texto son excelentes candidatos para completions. Del mismo modo, nos gustaría poder descomponer los párrafos en oraciones individuales para que podamos usar una oración en particular como prompt y el texto de las próximas cinco o diez oraciones que le siguen como su finalización. Esto significa que nos gustaría tener algún código en nuestra aplicación que capture el texto y lo divida en cosas como encabezados y oraciones.

Abordemos el segundo de estos problemas primero. Comencemos con un texto simple. En este ejemplo, tenemos un título y un párrafo de muestra de texto. Lo que queremos hacer es alimentar este texto en una función que lo divida en oraciones individuales. Luego podemos usar algunas de las oraciones como indicaciones. Digamos que cada tercera oración será una indicación y luego querremos usar los fragmentos de texto restantes entre las indicaciones como nuestras *completions*. Entonces, tomemos la primera oración resaltada en morado como nuestra primera indicación. GPT-3 se puede ajustar finamente y luego las dos siguientes oraciones en amarillo son el texto de la *completions* que sigue.

Luego repetiremos esto con la cuarta oración y las dos oraciones que siguen, y finalmente con la séptima oración y la única oración que sigue. Esto nos da una muestra de señal de entrada y completado de alta resolución en el texto elegido. Podemos agregar a esto una señal de entrada y completado de baja resolución tomando el texto del título como nuestra señal de entrada y el párrafo completo de ocho oraciones que sigue como nuestro completado. Esto nos dará un total de cuatro señales de entrada y cuatro completados de esta muestra de texto. Así que necesitaremos escribir algún código para hacer esto. Por favor, no mire si el código de Python le hace sentir incómodo.

En primer lugar, aquí hay una función en Python que tomará una cadena de texto como entrada y la dividirá en sus oraciones componentes y devolverá una lista de todas las oraciones en el texto. La magia de Python aquí es esta expresión regular. Esto verifica un punto completo, un signo de interrogación o un signo de exclamación seguido de uno o más espacios. Utiliza este delimitador para dividir la cadena en una lista de todas las oraciones y devuelve esta lista como la salida de la función.

Ahora, eso es un gran comienzo, pero necesitamos más que una simple lista de cadenas. Queremos poder tomar la enésima oración y usarla como nuestro prompt, y luego usar todas las oraciones intermedias como completado hasta que lleguemos al próximo prompt. Es posible que nos gustaría variar con qué frecuencia generamos prompts. ¿Será cada tres oraciones en el ejemplo que acabamos de ver, o quizás queremos generar prompts menos frecuentes para completados más largos, como un prompt cada 5 o 10 oraciones? Para hacer eso, debemos pasar una variable a la función que determinará con qué frecuencia generamos nuestros prompts.

Entonces, aquí está la función. Ahora está haciendo un poco más, por lo que parece un poco más intimidante para los que no son pythonistas, pero vamos a desglosarlo. Una vez que el texto se divide en oraciones (hemos visto cómo hacer eso), la función los agrupa en prompts y completados. Un prompt es simplemente la primera oración de cada grupo, mientras que el completado son las oraciones restantes en el grupo. La función crea dos listas, una para prompts (haciendo cortes en las oraciones y tomando la enésima oración en la lista) y luego otra lista para completados (recorriendo las oraciones intermedias y agregándolas a una lista de completados). Finalmente, la función crea un diccionario que en Python es un conjunto de pares de nombre-valor que asigna cada prompt a su correspondiente completado y utiliza algo llamado la biblioteca de pandas para convertir este diccionario en algo llamado un data frame. Piense en un data frame como una estructura de datos tabular en Python similar a una hoja de cálculo de Excel. El data frame resultante contiene dos columnas, una para prompts y otra para completados, lo que hace que sea fácil alimentar los datos en un modelo de lenguaje para afinarlos o para otras aplicaciones.

Si ejecutamos este código en una cadena de texto ficticia que contiene 50 oraciones, podemos ver que ha tomado cada quinta oración como una indicación y las oraciones intermedias como completadas. En este punto, estamos prácticamente terminados, francamente, este código funcionará bien para la mayoría de los propósitos, pero para hacerlo un poco más robusto, agregaremos algunas comprobaciones de errores simples y algunos métodos para mostrar los resultados de esta función en un archivo Json, un archivo CSV y una hoja de cálculo de Excel. Podemos reunir todo esto en una clase simple. Para inicializar un objeto de esta clase, puede pasar la cadena a dividir y un número que represente con qué frecuencia crearemos una indicación, digamos cada quinta or décima oración. Tenemos nuestra función familiar para dividir nuestra oración, en esta versión lo hemos hecho un poco más robusto con algunas comprobaciones de errores simples. Finalmente, hemos agregado tres funciones al final que crearán primero un archivo de Excel, segundo un archivo CSV y finalmente una cadena de Json de nuestro marco de datos de pandas. Luego podríamos escribir un breve script para crear tal objeto. Como entradas, pasamos una cadena y el número cinco para indicar que nos gustaría que cada quinta oración sea una indicación. Cuando ejecutamos este código, podemos ver la hoja de cálculo generada que contiene nuestras indicaciones y completadas creadas a partir del texto que suministramos. Entonces hemos resuelto el problema de dividir nuestro texto. Lo siguiente que debemos resolver es poder obtener texto de Internet. Para hacer esto, vamos a usar un paquete popular de Python llamado Beautiful Soup. Cuando ve una página web en un navegador, hace una solicitud a un servidor web y envía de vuelta algo de HTML. Esto contiene todo el contenido y el formato para que su navegador pueda mostrarlo. Beautiful Soup le permite tomar los componentes individuales del HTML y, en lugar de mostrarlos en un navegador, se devuelven a su programa Python. Aquí hay un ejemplo de cómo funciona. Asegúrese de que el sitio web que visita y utiliza Beautiful Soup no le importe que se raspe su contenido. Un sitio web llamado "Quotes to Scrape" es una apuesta bastante segura. Aquí hay un simple script de Python que utiliza Beautiful Soup para visitar este sitio y recuperar el HTML. Luego usa Beautiful Soup para mostrar los datos de nuestro programa. Cuando lo ejecutamos, podemos ver la primera cita de Einstein. Ahora modifiquemos el script un poco para que podamos ver más del contenido de la página. En ambos casos, podemos obtener el contenido que necesitamos como una cadena de texto. Recuerde que podemos usar esta cadena como entrada en nuestra clase de división descrita anteriormente. Finalmente, veamos la página de Wikipedia para Beautiful Soup. Raspar esta página puede ser un poco como un chef usando su propio libro de recetas para cocinar la cena o un peluquero cortando su propio cabello. Pero a veces solo tienes que abrazar lo meta y dejar que la belleza de Beautiful Soup haga su trabajo. En este video, hemos continuado nuestra discusión sobre el ajuste fino de las redes neuronales Transformer. Hablamos sobre por qué el ajuste fino es importante y cómo se puede utilizar para crear modelos personalizados.

Discutimos métodos básicos de ingeniería de prompts y manipulación de texto simples para construir pipelines automatizados para la afinación fina. El método más sencillo es llenar manualmente una hoja de cálculo de Excel con dos columnas: una llamada "prompts" y la otra llamada "completions", pero esto no es escalable. Por favor, considera esto sólo como último recurso. El internet está lleno de contenido y podemos usar herramientas para manipular texto y recuperar pasajes relevantes de internet. Identificamos dos problemas específicos que necesitan ser resueltos: uno es encontrar texto relevante en internet y el otro es dividirlo en prompts y completions. Vimos cómo resolver el segundo problema construyendo una clase separadora en Python. Para resolver el primer problema, presentamos Beautiful Soup. Esta popular biblioteca de Python permite a los usuarios extraer componentes HTML de una página web en lugar de mostrarlos en un navegador. Vimos cómo usar Beautiful Soup para extraer texto de un sitio web, como citas para raspar o la página de Wikipedia para Beautiful Soup. Por último, enfatizamos la importancia de afinar los modelos de IA para lograr multiplicadores de eficiencia y escalabilidad. Soy Richard Walker de Lucidate. Gracias por ver. Por favor, únete a mí en el próximo episodio donde veremos formas más sofisticadas de construir prompts y completions utilizando APIs de noticias, audio y video.