Fine-Tuning in OpenAI: Personalizzazione dei Modelli AI

Benvenuti a questo corso introduttivo sul fine-tuning dei modelli OpenAI. In questo percorso didattico esploreremo come personalizzare i modelli linguistici, incluso il recente GPT-4.1-nano (openai v 1.87), per adattarli alle vostre esigenze specifiche.

Impareremo a preparare i dati, caricarli sulla piattaforma OpenAI, avviare e monitorare i processi di fine-tuning, fino ad arrivare all'utilizzo pratico dei modelli personalizzati nelle vostre applicazioni.

Il fine-tuning rappresenta un potente strumento per migliorare le prestazioni dei modelli AI in contesti specifici. Con GPT-4.1-nano, OpenAI offre un equilibrio ottimale tra efficienza e prestazioni. Iniziamo insieme questo viaggio!

CM **da Claudio Marfia**

Cos'è il Fine-Tuning in OpenAI

Il fine-tuning è un processo che permette di personalizzare un modello linguistico pre-addestrato come GPT-3.5 utilizzando un proprio dataset specifico. È come insegnare nuove competenze a un modello già esperto.

Tecnicamente, il fine-tuning consiste nell'ulteriore addestramento di un modello esistente su un insieme di dati rappresentativi del compito specifico che si desidera eseguire, mantenendo le capacità generali acquisite durante l'addestramento originale.

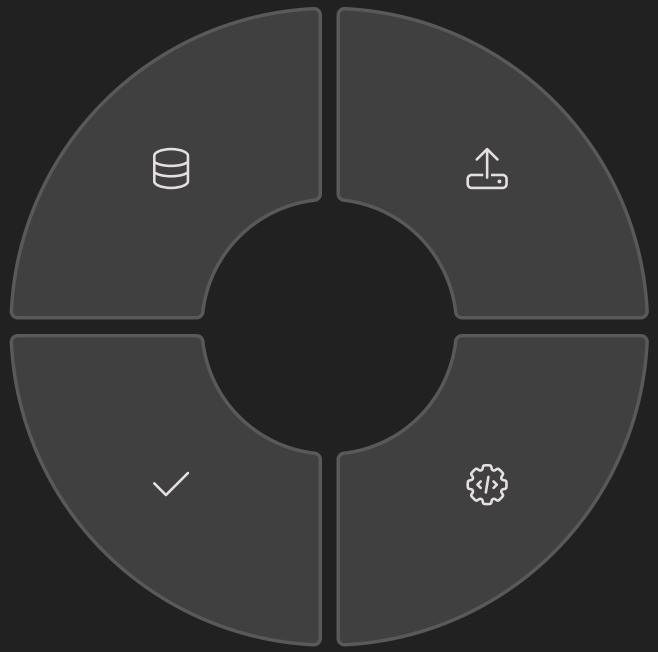
Questo processo consente di ottenere risposte più precise e coerenti per applicazioni specializzate, migliorando significativamente le prestazioni in contesti specifici come assistenza clienti, analisi di documenti tecnici o generazione di contenuti specialistici.

Raccolta Dati

Preparazione di esempi di input e output desiderati in formato specifico

Upload

Caricamento del dataset sulla piattaforma OpenAI



Utilizzo

Implementazione del modello personalizzato nelle applicazioni

Addestramento

Esecuzione del processo di fine-tuning sul modello base

Quando Utilizzare il Fine-Tuning con GPT-4.1-nano

Il fine-tuning di GPT-4.1-nano (openai v 1.87) è particolarmente utile quando avete bisogno di risposte consistenti e prevedibili in un dominio specifico. È indicato quando disponete di numerosi esempi di coppie input-output che rappresentano il comportamento desiderato per questo modello più leggero.

Comportamento costante e specifico

Quando è necessario che GPT-4.1-nano risponda in modo uniforme e prevedibile a richieste simili, mantenendo uno stile coerente anche con risorse computazionali limitate.

Conoscenza specializzata

Per domini altamente tecnici o di nicchia dove la conoscenza generale del modello base GPT-4.1-nano potrebbe beneficiare di ulteriore specializzazione.

Ottimizzazione dei costi

GPT-4.1-nano offre già efficienza, ma con il fine-tuning si ottiene un'ulteriore riduzione della lunghezza dei prompt necessari, con conseguente risparmio sui token utilizzati e minori costi operativi.

Stile di risposta personalizzato

Quando si desidera che GPT-4.1-nano adotti uno stile comunicativo specifico, coerente con il brand o le esigenze aziendali, pur mantenendo la leggerezza del modello.

GPT-4.1-nano (OpenAI v 1.87)

Caratteristiche Principali

- Versione ottimizzata e compatta di GPT-4
- Elevata efficienza computazionale
- Latenza ridotta rispetto ai modelli più grandi
- Costi operativi inferiori
- Integrazione semplificata in applicazioni

Casi d'Uso Ideali

- Applicazioni in tempo reale con vincoli di latenza
- Sistemi embedder per motori di ricerca semantica
- Assistenti virtuali leggeri per dispositivi con risorse limitate
- Elaborazione di testo in applicazioni mobile
- Automazione di processi aziendali su larga scala

GPT-4.1-nano rappresenta un equilibrio ottimale tra prestazioni e risorse richieste. Sebbene non raggiunga la profondità di ragionamento dei modelli più grandi come GPT-4 Turbo, offre risposte rapide e di qualità per numerose applicazioni pratiche, con un consumo di risorse significativamente inferiore. È particolarmente adatto per implementazioni che richiedono scalabilità e efficienza.

Formato JSONL per i Training File

Per il fine-tuning di GPT-4.1-nano (openai v 1.87), OpenAI richiede che i dati siano formattati in JSONL (JSON Lines), dove ogni riga rappresenta un singolo esempio di addestramento in formato JSON. Ogni esempio deve contenere una coppia di campi "prompt" e "completion" che rappresentano rispettivamente l'input e l'output desiderato.

{"prompt": "Converti in python: Calcola la somma dei numeri da 1 a 10", "completion": "sum = 0\nfor i in range(1, 11):\n   sum += i\nprint(sum)"}
{"prompt": "Converti in python: Trova il massimo in una lista", "completion": "def trova_massimo(lista):\n   return max(lista)\n\n# Esempio d'uso\nlista = [3, 7, 2, 9, 1]\nprint(trova_massimo(lista))"}

È fondamentale che i file JSONL siano correttamente formattati, con ogni riga contenente un oggetto JSON valido. Gli errori di formattazione sono tra le cause più comuni di fallimento nel processo di fine-tuning con GPT-4.1-nano. Utilizzate strumenti di validazione JSON per verificare la correttezza del vostro file prima dell'upload.

Creazione dei Prompt per Fine-Tuning GPT-4.1-nano

La qualità del fine-tuning con GPT-4.1-nano (openai v 1.87) dipende fortemente dalla qualità dei prompt utilizzati. Un buon prompt per questo modello deve essere chiaro, specifico e rappresentativo del tipo di richieste che il modello dovrà gestire in produzione.

Chiarezza e Specificità

I prompt per GPT-4.1-nano devono essere chiari e contenere tutte le informazioni necessarie per generare la risposta desiderata.

Varietà e Copertura

Includere una vasta gamma di esempi che coprano i diversi casi d'uso e variazioni linguistiche compatibili con GPT-4.1-nano.

Equilibrio dei Dati

Bilanciare i tipi di richieste per evitare che GPT-4.1-nano sia sbilanciato verso certi comportamenti.

Qualità e Coerenza

Assicurarsi che le risposte siano di alta qualità e coerenti con lo stile desiderato per il modello GPT-4.1-nano.

Quando si lavora con GPT-4.1-nano (openai v 1.87), è consigliabile includere un marcatore di fine risposta nei completion, come "\n\nEND", per aiutare il modello a capire quando terminare la generazione. Questo è particolarmente utile per risposte di lunghezza variabile in questa versione del modello.

Validazione del Dataset

Prima di procedere con il fine-tuning, è essenziale validare il dataset per garantirne la qualità e l'idoneità. Con l'introduzione di GPT-4.1-nano (openai v 1.87), questo processo è stato ulteriormente ottimizzato.

Aspetti da Verificare

- Correttezza del formato JSONL
- Lunghezza adeguata di prompt e completion
- Distribuzione bilanciata degli esempi
- Assenza di duplicati o errori
- Coerenza stilistica delle risposte

Strumenti di Validazione con GPT-4.1-nano

OpenAI v 1.87 include strumenti avanzati per la validazione dei dataset. GPT-4.1-nano offre funzionalità migliorate per analizzare il dataset e suggerire ottimizzazioni, verificando automaticamente la formattazione, identificando potenziali problemi e fornendo statistiche dettagliate.

È consigliabile eseguire questa validazione prima di ogni tentativo di fine-tuning per evitare sprechi di risorse e ottenere risultati ottimali con il nuovo modello GPT-4.1-nano.

Upload del Dataset

Una volta preparato e validato il dataset, è necessario caricarlo sulla piattaforma OpenAI. Esistono due metodi principali per effettuare l'upload: tramite l'API OpenAI o utilizzando l'interfaccia da riga di comando (CLI). Per questo esempio utilizzeremo GPT-4.1-nano (openai v 1.87).

Upload via API Python

```
import openai

response = openai.File.create(
  file=open("training_data.jsonl", "rb"),
  purpose="fine-tune"
)
file_id = response.id
print(f"File ID: {file_id}")
```

Upload via CLI

```
openai api files.create \
  --file training_data.jsonl \
  --purpose fine-tune
```

Durante l'upload, il sistema effettua ulteriori controlli di validazione. In caso di file di grandi dimensioni, l'upload potrebbe richiedere alcuni minuti. È importante conservare l'ID del file restituito dal sistema, poiché sarà necessario per avviare il job di fine-tuning.

Creazione del Job di Fine-Tuning

Dopo aver caricato il dataset, è possibile avviare il processo di fine-tuning creando un job specifico. In questa fase si definiscono i parametri di addestramento e si seleziona il modello base da personalizzare.

1 Preparazione Parametri

Scegliere il modello base (gpt-4.1-nano o davinci), definire gli iperparametri come learning rate e numero di epoche.

2 Creazione Job via API

```
import openai
# Usando OpenAI v1.87
client = openai.OpenAI()

response = client.fine_tuning.create(
  training_file="file-abc123",
  model="gpt-4.1-nano",
  n_epochs=4
)

job_id = response.id
print(f"Job ID: {job_id}")
```

3 Creazione Job via CLI

```
openai api fine_tunes.create \
--training_file file-abc123 \
--model gpt-4.1-nano \
--n_epochs 4
```

È possibile specificare ulteriori parametri come validation_file per valutare le prestazioni durante l'addestramento, o batch_size per controllare la dimensione dei batch di addestramento. Utilizzando OpenAI v1.87, il fine-tuning con GPT-4.1-nano offre prestazioni migliorate rispetto alle versioni precedenti. Una scelta accurata dei parametri può influenzare significativamente la qualità del modello risultante.

Monitoraggio del Job di Fine-Tuning

Durante il processo di fine-tuning, è importante monitorare l'avanzamento per verificare che tutto proceda correttamente. OpenAI fornisce diversi strumenti per tenere traccia dello stato del job e delle metriche di addestramento utilizzando GPT-4.1-nano.

Controllo Stato via API

```
import openai
client = openai.OpenAI()

# Recupero stato di un job specifico
status = client.fine_tuning.jobs.retrieve("ftjob-abc123")
print(status)

# Lista di tutti i job attivi
jobs = client.fine_tuning.jobs.list()
for job in jobs.data:
    print(f"{job.id}: {job.status}")
```

Monitoraggio in Tempo Reale

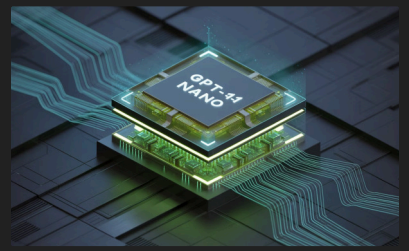
È possibile monitorare gli eventi in tempo reale utilizzando:

```
# Via API
events = client.fine_tuning.jobs.list_events(
   fine_tuning_job_id="ftjob-abc123"
)
print(events)

# Via CLI con streaming
openai api fine_tuning.jobs.stream ftjob-abc123
```

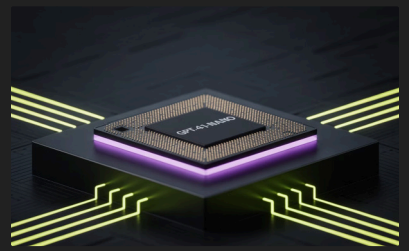
Il monitoraggio consente di identificare problemi come l'overfitting (quando il training loss continua a diminuire ma il validation loss aumenta) o altre anomalie durante l'addestramento con GPT-4.1-nano (openai v 1.87).

Differenza tra GPT-4.1-nano Base e Fine-Tuned



GPT-4.1-nano Base

- Conoscenza generale della versione 1.87
- Versatilità nelle risposte
- Richiede prompt dettagliati
- Latenza ridotta rispetto a modelli più grandi
- Bilanciamento tra prestazioni e dimensioni



GPT-4.1-nano Fine-Tuned

- Ottimizzato per compiti specifici su openai v 1.87
- Risposte coerenti e prevedibili
- Prompt più brevi ed essenziali
- Mantiene la velocità della versione base
- Miglior aderenza alle istruzioni specifiche del dominio

Il modello GPT-4.1-nano fine-tuned può essere considerato come una versione "addestrata" del modello base che ha imparato a seguire un pattern specifico di comportamento. Pur mantenendo i vantaggi di dimensioni ridotte e prestazioni efficienti della versione 1.87, la specializzazione avviene a scapito della versatilità generale: il modello potrebbe risultare meno efficace su compiti diversi da quelli per cui è stato ottimizzato.

Costi e Tempi per il Fine-Tuning

Il fine-tuning rappresenta un investimento sia in termini di tempo che di risorse economiche. È importante pianificare adeguatamente questi aspetti per ottimizzare il processo, specialmente quando si utilizza GPT-4.1-nano (OpenAI v 1.87).

2-4h

Tempo di Addestramento

La durata media di un job di fine-tuning per dataset di dimensioni moderate con GPT-4.1-nano (fino a 100MB).

$0.006

Costo per 1K Token

Costo approssimativo per addestramento su GPT-4.1-nano (OpenAI v 1.87), più economico rispetto a modelli precedenti.

x1.1-1.5

Costo Utilizzo

Moltiplicatore del costo di utilizzo rispetto al modello base per GPT-4.1-nano (efficienza migliorata).

I costi dipendono principalmente dalla dimensione del dataset, dal numero di epoche di addestramento e dal modello base scelto. Con GPT-4.1-nano (OpenAI v 1.87), si beneficia di tempi di addestramento più rapidi e costi inferiori rispetto alle versioni precedenti. È consigliabile iniziare con dataset più piccoli per test preliminari prima di investire in fine-tuning su larga scala. Inoltre, è importante considerare che anche con GPT-4.1-nano, i modelli fine-tuned hanno costi di utilizzo leggermente superiori rispetto ai modelli base.

Considerazioni Etiche e di Bias con GPT-4.1-nano

Il fine-tuning di GPT-4.1-nano (OpenAI v 1.87), pur essendo un potente strumento, comporta importanti considerazioni etiche. Questo modello più compatto tende comunque ad amplificare i bias presenti nei dati di addestramento, quindi è fondamentale prestare attenzione alla qualità e alla diversità del dataset utilizzato.

Rappresentatività del Dataset

Con GPT-4.1-nano, è cruciale assicurarsi che il dataset includa esempi diversificati che rappresentino adeguatamente tutte le categorie di utenti e casi d'uso. Nonostante le dimensioni ridotte del modello, un dataset sbilanciato porterà comunque a risultati che riflettono e potenzialmente amplificano questi squilibri.

Revisione delle Risposte

Data la natura più leggera di GPT-4.1-nano, è essenziale esaminare attentamente le risposte generate nei dati di addestramento per identificare e correggere contenuti potenzialmente problematici, discriminatori o inappropriati prima del fine-tuning, verificando che il modello più piccolo mantenga standard etici elevati.

Test su Scenari Critici

Testare il modello GPT-4.1-nano fine-tuned con input progettati specificamente per rilevare bias o comportamenti problematici, specialmente considerando che i modelli più piccoli potrebbero avere diverse sensibilità rispetto ai loro equivalenti più grandi in aree come discussioni su genere, etnia, politica o religione.

Documentazione e Trasparenza

Documentare le scelte effettuate durante la creazione del dataset e il processo di fine-tuning per GPT-4.1-nano (v 1.87), rendendo trasparenti non solo i potenziali limiti del modello per gli utenti finali, ma anche le differenze di prestazioni e comportamento rispetto ad altri modelli della famiglia GPT.

Versionamento e Gestione dei Modelli

Man mano che si creano più versioni di modelli fine-tuned, diventa essenziale implementare un sistema di versionamento e gestione efficace. Questo permette di tracciare le modifiche, confrontare le prestazioni e selezionare la versione più adatta per ogni caso d'uso.

Strategie di Versionamento

- Utilizzare convenzioni di denominazione consistenti
- Registrare metadati come dataset utilizzato e parametri
- Conservare i risultati dei test di valutazione
- Implementare un sistema di tagging per uso e performance

Gestione via API

```
# Utilizzo di GPT-4.1-nano con OpenAI v1.87
from openai import OpenAI

client = OpenAI()

# Elenco modelli disponibili
models = client.models.list()
for model in models.data:
    if "ft-" in model.id:
        print(model.id)

# Creazione richiesta con GPT-4.1-nano
response = client.chat.completions.create(
    model="gpt-4.1-nano",
    messages=[
        {"role": "system", "content": "Sei un assistente esperto."},
        {"role": "user", "content": "Descrivi questo modello fine-
tuned"}
    ]
)

# Eliminazione modello fine-tuned
client.models.delete("ft-abc123")
```

È buona pratica conservare i dataset utilizzati per ogni versione del modello, in modo da poter replicare il processo di fine-tuning se necessario. Inoltre, documentare i casi d'uso specifici e le prestazioni di ciascun modello aiuta a scegliere la versione più appropriata per ogni applicazione.

Testing del Modello Fine-Tuned e Limiti Attuali

Strategie di Testing

Dopo il completamento del fine-tuning, è essenziale testare rigorosamente il modello per verificarne le prestazioni. Utilizzate un set di test separato, non incluso nel dataset di addestramento, per valutare l'efficacia del modello in scenari reali.

- Test A/B confrontando modello base e fine-tuned
- Valutazione quantitativa con metriche oggettive
- Feedback qualitativo da utenti reali
- Test di stress con casi limite e input inattesi

Per progetti che richiedono efficienza e costi contenuti, considerate l'utilizzo di GPT-4.1-nano (openai v 1.87) per i test iniziali prima di passare a modelli più grandi.

Limiti Attuali

È importante essere consapevoli dei limiti attuali del fine-tuning sui modelli GPT-3.5/4:

- Impossibilità di aggiungere nuove conoscenze fattuali
- Necessità di dataset relativamente grandi (almeno 50-100 esempi)
- Tempo di attesa per la disponibilità del modello fine-tuned
- Limitata trasferibilità tra versioni di modelli
- Minore flessibilità rispetto al modello base per compiti generici
- Costi più elevati per modelli molto specializzati

I modelli più leggeri come GPT-4.1-nano offrono un buon compromesso tra performance e costi, ma potrebbero mostrare capacità ridotte rispetto alle versioni complete.

Nonostante questi limiti, il fine-tuning rimane uno strumento potente per creare applicazioni AI specializzate. Le sue capacità continuano ad evolversi con ogni aggiornamento della piattaforma OpenAI, offrendo nuove possibilità per personalizzare l'intelligenza artificiale secondo le vostre esigenze specifiche.