25/12/2025/H17:53:59

OpenAI publie la version bêta de son SDK Python, et offre ainsi un accès pratique à l'API OpenAI à partir des applications écrites en Python

OpenAI publie la version bêta de son SDK Python, et offre ainsi un accès pratique à l'API OpenAI à partir des applications écrites en Python

```
OpenAI a dévoilé la version bêta de son SDK Python, marquant ainsi une étape importante dans l'amélioration de l'accès à l'API OpenAI pour les développeurs Python. La bibliothèque OpenAI Python offre un moyen simplifié pour les applications basées sur Python d'interagir avec l'API OpenAI, tout en offrant une opportunité de tests et de retours avant le lancement officiel de la version 1.0.
```

La bibliothèque OpenAI Python fournit un accès pratique à l'API OpenAI à partir d'applications écrites en langage Python. Elle comprend un ensemble prédéfini de classes pour les ressources API qui s'initialisent dynamiquement à partir des réponses API, ce qui la rend compatible avec un large éventail de versions de l'API OpenAI. Vous trouverez des exemples d'utilisation de la bibliothèque OpenAI Python dans la référence API et dans l'OpenAI Cookbook. Installation Pour commencer, assurez-vous d'avoir Python 3.7.1 ou une version plus récente. Si vous souhaitez simplement utiliser le paquetage, exécutez : pip install –upgrade openai Après avoir installé le paquetage, importez-le au début d'un fichier : Pour installer ce paquetage à partir des sources afin d'y apporter des modifications, exécutez la commande suivante à partir de la racine du dépôt : Dépendances optionnelles Installer les dépendances pour openai.embeddings_utils : pip install openai[embeddings] Installer le support pour Weights & Biases : pip install openai[wandb] Les bibliothèques de données comme numpy et pandas ne sont pas installées par défaut en raison de leur taille. Elles sont nécessaires pour certaines fonctionnalités de cette bibliothèque, mais généralement pas pour communiquer avec l'API. Si vous rencontrez une MissingDependencyError , installez-les avec : pip install openai[datalib] Utilisation La bibliothèque doit être configurée avec la clé secrète de votre compte, disponible sur le site web. Définissez-la comme variable d'environnement OPENAI_API_KEY avant d'utiliser la bibliothèque : export OPENAI_API_KEY='sk-...' Ou définir openai.api_key à sa valeur : openai.api_key = “sk-...” Des exemples d'utilisation de cette bibliothèque pour accomplir diverses tâches peuvent être trouvés dans l'OpenAI Cookbook. Il contient des exemples de code pour : la classification en utilisant le réglage fin, le clustering, la recherche de code, la personnalisation des embeddings, la réponse aux questions à partir d'un corpus de documents, les recommandations, la visualisation des embeddings, et plus encore. La plupart des points de terminaison prennent en charge un paramètre request_timeout. Ce paramètre prend une Union[float, Tuple[float, float]] et lèvera une erreur openai.error.Timeout si la requête dépasse ce temps en secondes (Voir : https://re...tart/#timeouts). Complétions du chat Les modèles de chat tels que gpt-3.5-turbo et gpt-4 peuvent être appelés à l'aide du endpoint “chat completions”. 1 2 completion = openai.ChatCompletion.create(model=“gpt-3.5-turbo”, messages=[{“role”: “user”, “content”: “Hello

world”}]) print(completion.choices[0].message.content) Complétions Les modèles de texte tels que babbage-002 ou davinci-002 (et les anciens modèles de complétions) peuvent être appelés à l'aide du endpoint des complétions. 1 2 completion = openai.Completion.create(model=“davinci-002”, prompt=“Hello world”) print(completion.choices[0].text) Incorporations Les incorporations sont conçus pour mesurer la similarité ou la pertinence entre les chaînes de texte. Pour obtenir une incorporation pour une chaîne de texte, vous pouvez utiliser ce qui suit : 1 2 3 4 5 text_string = “sample text” model_id = “text-embedding-ada-002” embedding = openai.Embedding.create(input=text_string, model=model_id)['data'][0]['embedding'] Réglage fin Le réglage fin d'un modèle sur des données d'entraînement peut à la fois améliorer les résultats (en donnant au modèle plus d'exemples pour apprendre) et réduire le coût/la latence des appels d'API en réduisant la nécessité d'inclure des exemples d'entraînement dans les prompts. 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 # Create a fine-tuning job with an already uploaded file openai.FineTuningJob.create(training_file=“file-abc123”, model=“gpt-3.5-turbo”) # List 10 fine-tuning jobs openai.FineTuningJob.list(limit=10) # Retrieve the state of a fine-tune openai.FineTuningJob.retrieve(“ft-abc123”) # Cancel a job openai.FineTuningJob.cancel(“ft-abc123”) # List up to 10 events from a fine-tuning job openai.FineTuningJob.list_events(id=“ft-abc123”, limit=10) # Delete a fine-tuned model (must be an owner of the org the model was created in) openai.Model.delete(“ft:gpt-3.5-turbo:acemeco:suffix:abc123”) Pour enregistrer les résultats de l'entraînement à partir du réglage fin dans Pondérations et biais, utilisez : Modération OpenAI fournit un endpoint de modération gratuit qui peut être utilisé pour vérifier si le contenu est conforme à la politique de contenu d'OpenAI. moderation_resp = openai.Moderation.create(input=“Here is some perfectly innocuous text that follows all OpenAI content policies.”) Génération d'images (DALL-E) DALL-E est un modèle d'image génératif qui peut créer de nouvelles images sur la base d'une invite. image_resp = openai.Image.create(prompt=“two dogs playing chess, oil painting”, n=4, size=“512×512”) Audio (Whisper) L'API de conversion de la parole en texte fournit deux endpoints, les transcriptions et les traductions, basées sur le modèle Whisper large-v2 à source ouverte. 1 2 3 4 f = open(“path/to/file.mp3”, “rb”) transcript = openai.Audio.transcribe(“whisper-1”, f) transcript = openai.Audio.translate(“whisper-1”, f) API Async La prise en charge d'Async est disponible dans l'API en ajoutant a à une méthode liée au réseau : 1 2 async def create_chat_completion(): chat_completion_resp = await openai.ChatCompletion.acreate(model=“gpt-3.5-turbo”, messages=[{“role”: “user”, “content”: “Hello world”}]) Pour rendre les requêtes asynchrones plus efficaces, vous pouvez passer votre propre aiohttp.ClientSession , mais vous devez fermer manuellement la session client à la fin de votre programme/boucle d'événements : 1 2 3 4 5 from aiohttp import ClientSession openai.aiosession.set(ClientSession()) # At the end of your program, close the http session await openai.aiosession.get().close() Interface en ligne de commande Cette bibliothèque fournit également un utilitaire de ligne de commande openai qui permet d'interagir facilement avec l'API depuis votre terminal. Exécutez openai api -h pour l'utiliser. 1 2 3 4 5 6 7 8 9 10 11 12 13 14 # list models openai api models.list # create a chat completion (gpt-3.5-turbo, gpt-4, etc.) openai api chat_completions.create -m gpt-3.5-turbo -g user “Hello world” # create a completion (text-davinci-003, text-davinci-002, ada, babbage, curie, davinci, etc.) openai api completions.create -m ada -p “Hello world” # generate images via DALL·E API openai api image.create -p “two dogs playing chess, cartoon” -n 1 # using openai through a proxy openai –proxy=http://proxy.com api models.list Endpoints Microsoft Azure Pour utiliser la bibliothèque avec les endpoints Microsoft Azure, vous devez définir l'api_type, l'api_base et l'api_version en plus de l'api_key. L'api_type doit être défini à “azure” et les autres correspondent aux propriétés de votre endoint. En outre, le nom du déploiement doit être transmis en tant que paramètre du moteur. 1 2 3 4 5 6 7 8 9 10 11 import openai openai.api_type = “azure” openai.api_key = “…” openai.api_base = “https://example-endpoint.openai.azure.com” openai.api_version = “2023-05-15” # create a chat completion chat_completion = openai.ChatCompletion.create(deployment_id=“deployment-name”, model=“gpt-3.5-turbo”, messages=[{“role”: “user”, “content”: “Hello world”}]) # print the

completion print(chat_completion.choices[0].message.content) Veuillez noter que pour le moment, les endpoints Microsoft Azure ne peuvent être utilisés que pour les opérations d'achèvement, d'incorporation et de réglage fin. Authentification Microsoft Azure Active Directory Afin d'utiliser Microsoft Active Directory pour authentifier votre endpoint Azure, vous devez définir l'api_type à “azure_ad” et passer le jeton d'authentification acquis à api_key. Les autres paramètres doivent être définis comme indiqué dans la section précédente. 1 2 3 4 5 6 7 8 9 10 11 12 from azure.identity import DefaultAzureCredential import openai # Request credential default_credential = DefaultAzureCredential() token = default_credential.get_token(“https://cognitiveservices.azure.com/.default”) # Setup parameters openai.api_type = “azure_ad” openai.api_key = token.token openai.api_base = “https://example-endpoint.openai.azure.com/” openai.api_version = “2023-05-15” Source : Python SDK Et vous ? Qu'en pensez-vous ? Quelles fonctionnalités trouvez-vous intéressantes ? Voir aussi OpenAI permettra aux développeurs d'intégrer ChatGPT dans leurs applications via une API, mais l'outil est-il prêt à être utilisé dans des environnements de production ? OpenAI permet désormais aux développeurs d'apporter leurs propres données pour personnaliser GPT-3.5 Turbo, afin de créer et d'exécuter des modèles plus performants pour leurs cas d'utilisation OpenAI annonce la disponibilité générale de son API GPT-4, permettant ainsi aux développeurs d'intégrer la dernière génération de son IA générative dans leurs applications https://intelligence-artificielle.developpez.com/actu/349337/OpenAI-publie-la-version-beta-de-son-SDK-Python-et-offre-ainsi-un-acces-pratique-a-l-API-OpenAI-a-partir-des-applications-ecrites-en-Python/