Messages de Commit Git Propulsés par l'IA

Ce script Python doit être placé dans un répertoire inclus dans le PATH de votre système, comme `~/bin`.

```
import subprocess
import os
from openai import OpenAI
from dotenv import load_dotenv
import argparse

load_dotenv()

def gitmessageai(push=True, only_message=False):
    # Stage tous les changements
    subprocess.run(["git", "add", "-A"], check=True)

    # Obtenir le diff des changements stagés
    diff_process = subprocess.run(["git", "diff", "--staged"], capture_output=True, text=True, check=Tru
    diff = diff_process.stdout

    if not diff:
        print("Aucun changement à committer.")
        return

    # Préparer le prompt pour l'IA
    prompt = f"""
Génère un message de commit concis au format Conventional Commits pour les changements de code suivants
Utilise l'un des types suivants : feat, fix, docs, style, refactor, test, chore, perf, ci, build, ou re
Si applicable, inclut une portée entre parenthèses pour décrire la partie du codebase affectée.
Le message de commit ne doit pas dépasser 70 caractères.
```

```
Changements de code :
{diff}

Message de commit :
"""

    # Envoyer le prompt à l'API DeepSeek
    api_key = os.environ.get("DEEPSEEK_API_KEY")
    if not api_key:
        print("Erreur : La variable d'environnement DEEPSEEK_API_KEY n'est pas définie.")
        return

    client = OpenAI(api_key=api_key, base_url="https://api.deepseek.com")



    try:
        response = client.chat.completions.create(
            model="deepseek-chat",
            messages=[
                {"role": "user", "content": prompt}
            ],
            max_tokens=100
        )
        if response and response.choices:
            commit_message = response.choices[0].message.content.strip()
            commit_message = commit_message.replace('`', '')
        else:
            print("Erreur : Aucune réponse de l'API.")
            return
    except Exception as e:
        print(f"Erreur lors de l'appel à l'API : {e}")
        return

    # Vérifier si le message de commit est vide
    if not commit_message:
        print("Erreur : Message de commit vide généré. Abandon du commit.")
```

```
        return

    if only_message:
        print(f"Message de commit suggéré : {commit_message}")
        return

    # Committer avec le message généré
    subprocess.run(["git", "commit", "-m", commit_message], check=True)

    # Pousser les changements
    if push:
        subprocess.run(["git", "push"], check=True)
    else:
        print("Changements commités localement, mais non poussés.")

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Génère un message de commit avec l'IA et committe les
    parser.add_argument('--no-push', dest='push', action='store_false', help='Commite les changements l
    parser.add_argument('--only-message', dest='only_message', action='store_true', help='Affiche uniqu
    args = parser.parse_args()
    gitmessageai(push=args.push, only_message=args.only_message)
```

Ensuite, dans votre fichier `~/.zprofile`, ajoutez ce qui suit :

```
alias gpa='python ~/bin/gitmessageai.py'
alias gca='python ~/bin/gitmessageai.py --no-push'
alias gm='python ~/bin/gitmessageai.py --only-message'
```