Facebook Auto Poster - Bot Telegram

Description

Le Facebook Auto Poster est un bot Telegram qui permet aux utilisateurs d'automatiser la publication de contenu sur leurs pages Facebook. Il offre une solution simple et efficace pour générer du contenu à l'aide de l'IA et le publier sur Facebook à intervalles réguliers.

Fonctionnalités principales:

- Connexion sécurisée aux pages Facebook via OAuth
- Génération automatique de contenu engageant avec OpenAI (GPT-4o-mini)
- Publication à la demande ou automatisée à intervalles réguliers
- Personnalisation des thèmes et intervalles de publication
- Gestion des tokens d'accès Facebook (avec alertes d'expiration)
- Interface utilisateur Telegram intuitive avec boutons interactifs

Prérequis

- Python 3.8+
- Compte développeur Facebook avec une application configurée
- Clé API OpenAI
- Token de bot Telegram

Installation

1. Clonez le dépôt:

bash

```
git clone https://github.com/votreusername/facebook-auto-poster.git
cd facebook-auto-poster
```

2. Créez et activez un environnement virtuel:

bash

```
python -m venv venv
source venv/bin/activate  # Sur Windows: venv\Scripts\activate
```

3. Installez les dépendances:

bash

```
pip install -r requirements.txt
```

4. Créez un fichier `.env` à la racine du projet avec les variables d'environnement suivantes:

```
TELEGRAM_TOKEN=your_telegram_bot_token
FACEBOOK_APP_ID=your_facebook_app_id
FACEBOOK_APP_SECRET=your_facebook_app_secret
OPENAI_API_KEY=your_openai_api_key
ADMIN_TELEGRAM_ID=your_telegram_id_for_admin_alerts
REDIRECT_URI=https://yourusername.pythonanywhere.com/facebook_callback
```

Structure du projet

```
facebook-auto-poster/
├── main.py                # Fichier principal du bot
├── .env                   # Variables d'environnement (à créer)
├── requirements.txt       # Dépendances
├── messages.csv           # Historique des messages publiés
├── users.csv              # Configuration des utilisateurs
└── images/                # Dossier pour les images utilisées dans les publications
```

Configuration Facebook

1. Créez une application sur Facebook for Developers
2. Configurez l'authentification OAuth avec les permissions suivantes:
   - `pages_show_list`
   - `pages_read_engagement`
   - `pages_manage_posts`
   - `pages_manage_metadata`
3. Ajoutez l'URL de redirection `https://yourusername.pythonanywhere.com/facebook_callback` dans la configuration de l'application Facebook
4. Notez l'ID de l'application et le secret pour les variables d'environnement

Déploiement sur PythonAnywhere

Pour déployer ce bot sur PythonAnywhere, suivez ces étapes:

1. **Créez un compte** sur PythonAnywhere
2. **Créez une nouvelle application web**:

- Allez dans la section "Web" du tableau de bord
- Cliquez sur "Add a new web app"
- Choisissez "Manual configuration" et sélectionnez Python 3.8+
- Définissez le chemin vers votre application Flask

3. **Configurez votre application Flask**:
   - Créez un fichier `flask_app.py` dans votre répertoire principal:

```python
from flask import Flask, request, redirect
import os
import requests
import json

app = Flask(__name__)

@app.route('/facebook_callback')
def facebook_callback():
    code = request.args.get('code')
    state = request.args.get('state')

    # Rediriger vers Telegram avec le code
    telegram_url = f"https://t.me/votre_bot_telegram?start=code={code}&state={state}"
    return redirect(telegram_url)

if __name__ == '__main__':
    app.run(debug=True)
```

4. **Configurez le fichier WSGI**:
   - Modifiez le fichier WSGI généré automatiquement par PythonAnywhere pour qu'il pointe vers votre application Flask:

```python
import sys

path = '/home/yourusername/your-project-directory'
if path not in sys.path:
    sys.path.append(path)

from flask_app import app as application
```

5. **Créez un fichier requirements.txt**:

```
python-telegram-bot>=13.0
requests>=2.27.1
openai>=0.27.0
python-dotenv>=0.19.0
Flask>=2.0.0
```

6. **Installez les dépendances**:
   - Dans la console PythonAnywhere, exécutez:

   bash

   ```
   pip install --user -r requirements.txt
   ```

7. **Exécutez le bot** en arrière-plan:
   - Allez dans la section "Consoles" du tableau de bord PythonAnywhere
   - Créez une nouvelle console Bash
   - Naviguez vers votre répertoire de projet
   - Exécutez `nohup python main.py &` pour démarrer le bot en arrière-plan

8. **Configurez les tâches planifiées**:
   - Dans l'onglet "Tasks" de PythonAnywhere, créez une tâche quotidienne
   - Ajoutez une commande pour vérifier et redémarrer votre bot si nécessaire:

   bash

   ```
   cd ~/your-project-directory && ./check_and_restart.sh
   ```

9. **Création du script de redémarrage** (check_and_restart.sh):

   bash

   ```
   #!/bin/bash

   # Vérifier si le processus existe
   if ! pgrep -f "python main.py" > /dev/null
   then
       echo "Bot is not running. Restarting..."
       nohup python main.py > bot.log 2>&1 &
       echo "Bot restarted."
   else
       echo "Bot is already running."
   fi
   ```

   N'oubliez pas de rendre le script exécutable:

   bash

   ```
   chmod +x check_and_restart.sh
   ```

10. **Configuration du webhook** (si nécessaire):
    - Si votre application utilise des webhooks au lieu du polling, configurez-les dans la section Web de PythonAnywhere

Gérer l'authentification Facebook

Pour que l'authentification Facebook fonctionne correctement avec PythonAnywhere, vous devez:

1. **Configurer le système de callback**: Le bot utilise actuellement une simulation de callback dans le code original. Pour un déploiement en production sur PythonAnywhere, il faut implémenter un vrai système de callback via Flask.
2. **Flux d'authentification**:
    - L'utilisateur clique sur le bouton "Se connecter à Facebook" dans le bot Telegram
    - Il est redirigé vers l'authentification Facebook
    - Après autorisation, Facebook redirige vers votre URL de callback PythonAnywhere
    - L'application Flask capture le code et le state
    - L'utilisateur est redirigé vers Telegram avec ces paramètres
3. **Transmission du code**: Le défi est de transmettre le code d'autorisation de l'application web au bot. Deux approches sont possibles: a) **Redirection avec paramètres deep link**:

```python
# Dans flask_app.py
@app.route('/facebook_callback')
def facebook_callback():
    code = request.args.get('code')
    state = request.args.get('state')
    # Créer un lien deep avec les paramètres
    return redirect(f"https://t.me/votre_bot?start=code_{code}_state_{state}")
```

b) **Stockage temporaire et récupération**:

```python
# Dans flask_app.py
@app.route('/facebook_callback')
def facebook_callback():
    code = request.args.get('code')
    state = request.args.get('state')

    # Stocker temporairement dans un fichier ou une base de données
    save_auth_code(state, code)

    # Rediriger l'utilisateur vers Telegram
    return redirect(f"https://t.me/votre_bot?start=auth_complete")
```

4. **Sécurisation**:
    - Assurez-vous que votre application Flask utilise HTTPS (fourni par PythonAnywhere)
    - Validez toujours le paramètre "state" pour éviter les attaques CSRF
    - Ne stockez pas les codes d'autorisation plus longtemps que nécessaire

Utilisation

Démarrer le bot

```bash
python main.py
```

Commandes Telegram

- `/start` - Démarre le bot et affiche le menu principal

Fonctionnalités utilisateur

1. **Connexion Facebook**
    - Cliquez sur "Se connecter à Facebook" pour lier votre compte
    - Autorisez l'application à accéder à vos pages
    - Sélectionnez la page à utiliser pour les publications
2. **Contrôle des publications**
    - **Publier maintenant**: Génère et publie immédiatement un post
    - **Démarrer auto**: Active les publications automatiques selon l'intervalle configuré
    - **Arrêter auto**: Désactive les publications automatiques
    - **Statut**: Affiche l'état actuel de la configuration
3. **Paramètres**
    - **Changer le thème**: Personnalise le thème des publications générées
    - **Modifier l'intervalle**: Change la fréquence des publications automatiques (minimum 30 minutes)
    - **Reconnecter Facebook**: Renouvelle l'authentification Facebook

Gestion des tokens Facebook

Le bot inclut un système qui:

- Stocke les tokens d'accès de longue durée (60 jours)
- Surveille quotidiennement les tokens qui vont expirer
- Alerte les utilisateurs et l'administrateur lorsqu'un token expire dans moins de 2 jours

Génération de contenu

Le bot utilise le modèle GPT-4o-mini d'OpenAI pour générer des messages engageants qui:

- Commencent par un emoji pertinent
- Précisent que le service est gratuit
- Utilisent un ton amical et engageant
- Incluent un appel à l'action clair
- Intègrent un lien vers le bot Telegram de pronostics football
- Respectent le thème configuré par l'utilisateur

Stockage des données

Les données sont stockées dans deux fichiers CSV:

- `users.csv` - Configuration des utilisateurs, tokens et paramètres
- `messages.csv` - Historique des messages publiés

Dépendances principales

- `python-telegram-bot` - Interface avec l'API Telegram
- `requests` - Gestion des requêtes HTTP pour l'API Facebook
- `openai` - Intégration avec l'API OpenAI
- `python-dotenv` - Gestion des variables d'environnement

Sécurité

- Les tokens Facebook sont stockés localement et ne sont pas partagés
- L'authentification utilise le protocole OAuth standard de Facebook
- Les tokens d'accès sont renouvelés automatiquement via le processus de reconnexion

Dépannage

Problèmes courants

1. **Erreur lors de la connexion à Facebook**
   - Vérifiez que l'ID et le secret de l'application Facebook sont corrects
   - Assurez-vous que l'URL de redirection est configurée correctement dans Facebook
2. **Publications échouées**
   - Vérifiez que le token d'accès est valide et non expiré
   - Assurez-vous que les permissions de page sont correctes

3. **Erreurs de génération de contenu**
   - Vérifiez que la clé API OpenAI est valide et dispose de crédits suffisants
   - Assurez-vous que le thème configuré est approprié

Extension et personnalisation

Ajouter de nouvelles fonctionnalités

1. **Support pour plusieurs pages**
   - Modifiez la structure des données pour associer plusieurs pages à un utilisateur
   - Ajoutez des options de sélection de page dans l'interface
2. **Planification avancée**
   - Implémentez un système de calendrier pour des publications à des dates/heures spécifiques
   - Ajoutez une interface pour visualiser et gérer les publications planifiées
3. **Analyse des performances**
   - Intégrez l'API Facebook Insights pour récupérer des statistiques sur les publications
   - Créez des rapports périodiques sur l'engagement

Personnalisation du contenu généré

Modifiez la section `generate_ai_message()` pour adapter les instructions envoyées à l'API OpenAI selon vos besoins spécifiques.

Contribution

Les contributions sont les bienvenues! N'hésitez pas à ouvrir une issue ou à soumettre une pull request.

1. Forkez le dépôt
2. Créez une branche pour votre fonctionnalité (`git checkout -b feature/amazing-feature`)
3. Commitez vos changements (`git commit -m 'Add some amazing feature'`)
4. Poussez vers la branche (`git push origin feature/amazing-feature`)
5. Ouvrez une Pull Request

Licence

Ce projet est sous licence MIT. Voir le fichier `LICENSE` pour plus de détails.

Contact

Pour toute question ou assistance, veuillez créer une issue dans le dépôt GitHub.

Note: Ce bot est conçu pour la promotion d'un bot Telegram de pronostics football. Adaptez le contenu généré à vos propres besoins marketing en modifiant les instructions dans la fonction `generate_ai_message()`.