Déployer l'application CLIP sur un JupyterLab

Natacha Grim

septembre 2024

Refonte de l'application développée par ThorkildFregi et Altomator.

NOTA BENE : l'intégration d'une étape de *fine-tuning* est en développement.

L'application utilise le modèle CLIP (*Contrastive Language-Image Pre-training*) pour traiter un fonds d'images et l'interroger avec des requêtes textuelles, le tout depuis une interface web.

Contents

1 Cloner l'application

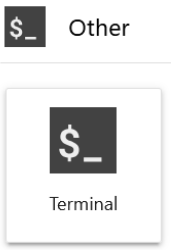
On va créer une copie de l'application, un "clone" sur un JupyterLab. Cela permettra de mettre à jour l'application très simplement.

Rendez-vous sur la page GitHub de l'application.

En haut de la page se trouve un bouton vert **code** :

Cliquez dessus et copiez l'url alors affichée :

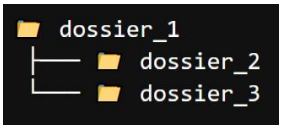
Dans l'interface Jupyter, ouvrez un nouvel onglet et sélectionnez **Terminal** :



1.1 Comment se déplacer depuis le terminal

À gauche de votre curseur clignotant permettant de saisir des commandes est affiché l'endroit où vous vous trouvez dans l'arborescence du serveur.

Au sein du terminal, on se déplace de dossier en dossier (*directory* en anglais) grâce à la commande `cd` (*change directory*). Par exemple :



On se trouve dans le `dossier_1`. Pour aller dans le `dossier_2`, il faut saisir :

```
cd dossier_2
```

Pour remonter d'un cran dans l'arborescence, il faut saisir la commande `cd ..`. On se trouve dans le `dossier_2`, pour aller dans le `dossier_3` on va saisir :

```
cd ..
cd dossier_3
```

Pour aller plus vite, on peut utiliser la touche de tabulation pour autocompléter le nom du dossier où on souhaite se déplacer. L'autocomplétion est partielle pour les noms de dossier qui commencent identiquement mais dont la fin est différente.

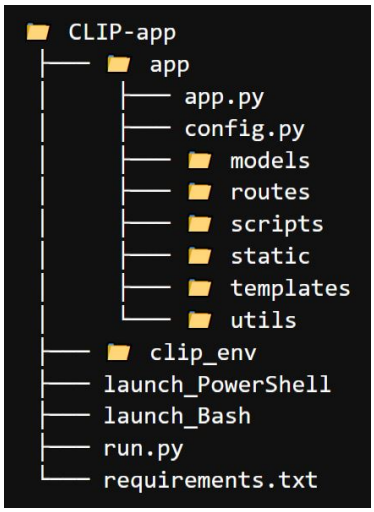
Pour l'exemple ci-dessus, saisir `cd do` puis appuyer sur la touche de tabulation va donner `cd dossier_`. Il faut alors saisir `2` ou `3` pour compléter le nom.

Déplacez-vous à l'endroit où vous voulez cloner l'application, saisissez la commande `git clone` puis collez l'URL copiée sur GitHub :

```
git clone <URL>
```

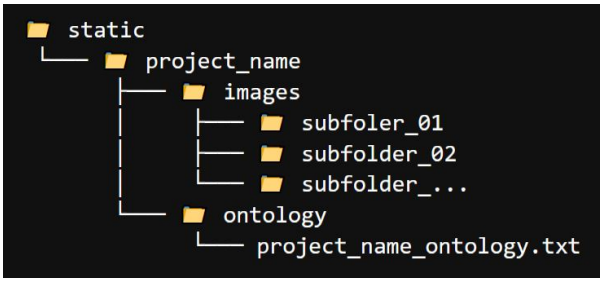
2 Structure

Pour utiliser l'application, il faut intégrer un fonds d'images et une ontologie dans sa structure. Ci-dessous, son architecture de base :



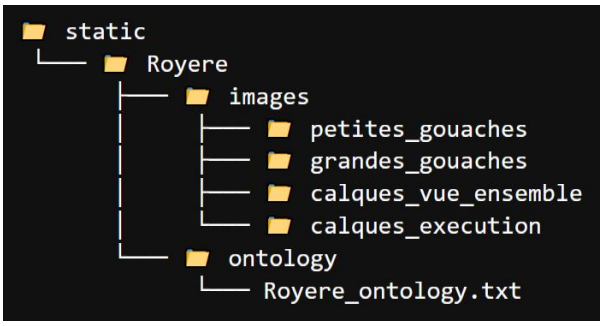
2.1 Dossiers et fichiers à créer

C'est dans le dossier `static` que vous chargez vos données. Vous devez créer les dossiers suivants en son sein :



Vous pouvez librement nommer le dossier `project_name` ainsi que chaque `subfolder` (les espaces, accents et caractères spéciaux sont proscrits).

En conséquence, le fichier `project_name_ontology.txt` doit être nommé d'après le nom du dossier `project_name`. Le nom des dossiers `images` et `ontology` doit rester inchangé:



2.2 Données requises

L'application a besoin de deux types de données pour fonctionner : des données visuelles (images) et des données textuelles (ontologie).

- Les images se trouvent dans le dossier `images` ;
- L'ontologie se trouve dans le fichier `project_name_ontology.txt` .

Le fichier d'ontologie `.txt` contient les mots-clés et leur description. La structure ci-dessous doit être respectée :

```
"Chaise": "siège à dossier et généralement sans bras"
"Tapis": "panneau d'étoffe, ouvrage que l'on pose sur un meuble, un mur ou un sol"
"Luminaire": "tout objet constituant l'éclairage et la décoration lumineuse"
```

2.3 Fichiers générés

Les scripts exécutés lors du lancement vont générer trois fichiers dans le dossier `project_name` :

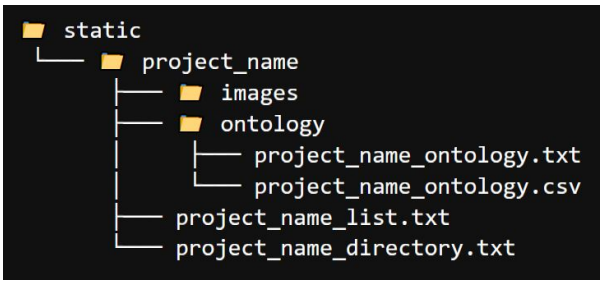
- `project_name_ontology.csv` ;
- `project_name_list.txt` ;
- `project_name_directory.txt` .

Le fichier `project_name_ontology.csv` résulte de la conversion du fichier `project_name_ontology.txt` au format `.csv` . Il sera utilisé par le modèle pour traiter les requêtes.

Le fichier `project_name_list.txt` liste les images du fonds et indique leur chemin relatif.

Le fichier `project_name_directory.txt` synthétise des informations sur le dossier `project_name` : son chemin relatif, son chemin absolu, le nombre de sous-dossiers et le nombre total d'images. Sur un pas de 10, le chemin d'une image est listé.

Une fois les scripts exécutés, la structure du dossier `project_name` est donc la suivante :



Le dernier fichier à être généré en amont du lancement de l'application se trouve dans le dossier `models`. Il s'agit d'un fichier contenant les représentations vectorielles des images (*embeddings* en anglais). Le temps nécessaire à la création de ce fichier dépend du nombre d'images chargées ainsi que de votre puissance de calcul.

Il est fondamental de ne pas interrompre l'exécution des scripts. Cela ne prendra du temps que pour le premier lancement car il ces fichiers n'ont besoin d'être générés qu'une seule fois.

3 Installer l'environnement virtuel

À la racine de l'application se trouve le dossier `clip_env`. Il s'agit d'un environnement virtuel.

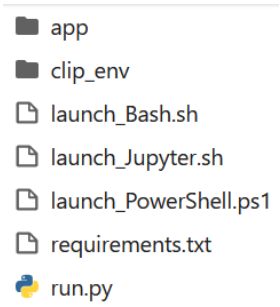
Pour faire fonctionner l'application, il faut installer des "paquets" sur lesquels reposent le code source. On parle alors de dépendances. Pour ce faire, on crée un environnement virtuel afin d'éviter les conflits avec d'autres paquets ou d'autres versions déjà présents sur l'ordinateur.

3.1 Étape 2 : créer l'environnement virtuel

Ouvrez votre terminal et déplacez-vous dans le dossier `CLIP_app` puis exécutez la commande suivante :

```
python3 -m venv clip_env
```

Un dossier `clip_env` devrait alors apparaître :



3.2 Étape 2 : installer les dépendances

Activez l'environnement virtuel :

```
source clip_env/bin/activate
```

Installez les dépendances contenues dans le fichier `requirements.txt` :

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

L'installation peut prendre plus ou moins de temps en fonction de votre vitesse de calcul et de la qualité de votre connexion. Vous devez attendre jusqu'à ce que

le chemin `/Application-CLIP/CLIP-app$` s'affiche à gauche de votre curseur.

Vous disposez maintenant d'un environnement virtuel. Lorsqu'il est activé, le chemin indiqué à gauche de votre curseur est précédé de l'indicateur `(clip_env)` .

Vous devrez l'activer à chaque lancement de l'application et le désactiver après chaque session d'utilisation.

L'activer :

```
source clip_env/bin/activate
```

Le désactiver :

```
deactivate
```

4 Lancement

Pour chaque lancement, il faut suivre les étapes ci-dessous :

4.1 Étape 1 : se positionner au bon endroit

Positionnez-vous dans le dossier `CLIP_app`.

4.2 Étape 2 : activer l'environnement virtuel

```
source clip_env/bin/activate
```

4.3 Étape 3 : lancer le *launcher* Jupyter

```
bash launch_Bash.sh
```

L'interface suivante devrait s'afficher :

Copiez l'URL affichée puis saisissez le nom de votre `project_name`. L'application va se lancer directement dans le terminal. Ouvrez alors un nouvel onglet dans votre navigateur et collez l'URL.

4.4 Étape 4 : quitter l'application

Depuis le terminal, pressez simplement la touche `ctrl` enfoncée et appuyez sur la touche `c`. N'oubliez pas de désactiver l'environnement virtuel.

5 Gestion des mises à jour

5.1 Mettre à jour l'application

Le clone de l'application reste en lien avec la version hébergée sur GitHub. Lorsque le code source est mis à jour, il faut manuellement "importer" les modifications pour mettre à niveau la version du clone.

Avant chaque lancement, il faut se déplacer dans le dossier `Application_CLIP` et exécuter les commandes suivantes :

```
git fetch
git pull
```

La commande `git fetch` vous permet de "scanner" la version du code sur GitHub pour savoir s'il y a eu des modifications.

La commande `git pull` vous permet de mettre à jour votre clone. L'exécution de cette commande doit afficher le texte `Déjà à jour`.

5.2 Mettre à jour l'ontologie

Si votre ontologie vient à évoluer, mettez simplement à jour le fichier `.txt`. Au prochain lancement de l'application, le fichier `.csv` se mettra à jour.