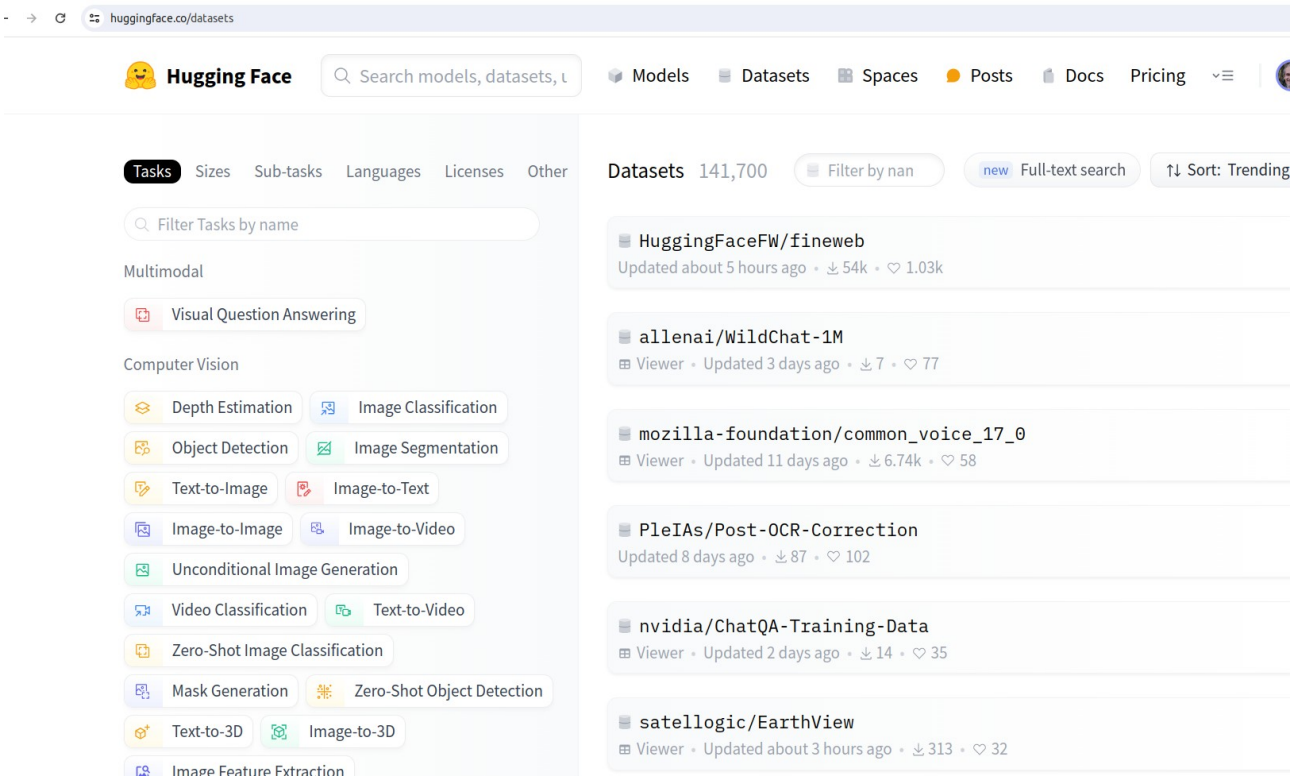
Saé Semestre 2 BDD

Contexte

Dans le domaine du traitement automatique du langage (TAL) et de l'intelligence artificielle (IA), les algorithmes d'apprentissage s'appuient sur des « Jeux de données » (en anglais dataset). Certains jeux de données permettent d'entraîner des modèles sur des tâches spécifiques (traduction, génération de paraphrase, discussion etc.).

Certains jeux de données sont mis à disposition du public. Par exemple le jeu de données PIAF (https://www.data.gouv.fr/fr/datasets/piaf-le-dataset-francophone-de-questions-reponses/#/resources) est mis en ligne sur le site gouvernemental OpenData. Le site communautaire HuggingFace (https://huggingface.co/) héberge aussi un grand nombre de jeux de données.



La page « dataset » du site HuggingFace.

Ces jeux de données sont distribués « en l'état », sans qu'il soit possible de connaître la qualité des données, c'est-à-dire si elles ont été validées, par qui etc. Dans ces « dataset », on ne sait donc pas vraiment ce qu'on va trouver et il faut faire « confiance ».

Objectif de la saé

L'objectif de la saé est de construire une application qui permette d'assurer la gestion du cycle de vie d'un jeu de données, ainsi que le suivi de sa qualité.

Les quelques paragraphes qui suivent précisent les attendus du travail à réaliser.

Quelques précisions sur les « jeux de données »

Un « jeu de données » est constitué d'un ensemble de données élémentaires de quelques milliers à quelques millions d'entrée. Une « entrée » peut par exemple être un couple de questions réponses.

Les jeux de données sont habituellement échangés sous forme de fichier au format JSON ou JSONL (le « L » signifiant une « Liste » de chaînes au format JSON).

Les jeux de données contiennent principalement du texte, mais ils peuvent aussi contenir des images (pour l'apprentissage de la reconnaissance d'images par exemple), des vidéos (par exemple pour la classification de vidés), des sons (pour l'entraînement à al reconnaissance d'image).

Les jeux de données d'apprentissage contiennent à la fois des éléments de contexte, une ou plusieurs instructions, et la réponse attendue. La structure des données n'est pas normalisée.

Fonctionnalités attendues de l'application

Votre application propose 3 grands groupes de fonctionnalité :

- visualisation des données.
  - On doit pouvoir afficher le contenu de la totalité du jeu de données
  - faire des recherches, sur la base d'une chaîne de caractères ou d'autres filtres à votre convenance
- modification et validation des données :
  - on doit pouvoir modifier le contenu d'une entrée si elle ne convient pas et la sauver en base
  - On doit pouvoir « valider » une entrée. La validation consiste, pour un opérateur humain, à affecter l'étiquette « validé » ou « non validé » à une entrée. Si c'est non validé, il doit y avoir une explication. Plusieurs validations doivent être possibles (une entrée peut avoir été validée par 1 personne et invalidée par 1 autre). Comme on ne peut pas valider l'ensemble du jeu de données en une seule fois, on privilégie une approche « statistique » : l'entrée à valider est tirée au hasard parmi toutes les entrées du jeu de donnée. On doit garder une trace d'une validation ou d'une invalidation (qui a validé ? Quand?)
  - on doit pouvoir créer une nouvelle entrée dans le jeu de données
- Une fonction d'import/export :
  - pour l'import, l'application doit permettre le chargement du fichier jsonl dans une base SQL
  - pou l'export, il doit être possible de reconstituer un fichier json à partir du contenu de l'intégralité de la base (à noter : tous les champs de la base n'ont pas à être exportés).

Ces fonctionnalités doivent pouvoir s'appliquer à un jeu de données existant ou à un nouveau jeu de données.

Un exemple de cas que l'on doit pouvoir conduire pour valider votre application :

- l'utilisateur charge un jeu de données au format jsonl
- l'utilisateur valide 10 entrées
- l'utilisateur ajoute 2 nouvelles entrée
- l'utilisateur utilise la fonction d'export, choisit « exporter uniquement les données validées », et un fichier jsonl est généré avec 12 données.

Option A : vous avez une base de données et une IHM pour chaque jeu de données. Votre application peut gérer uniquement un seul jeu de données, par exemple French-Alpaca-dataset-Instruct-110K. Si vous voulez traiter un jeu de données différent, il faut modifier l'interface graphique et la structure de la base de données. C'est le cas le plus simple.

Option B : votre IHM et votre base de données peuvent traiter n'importe quel jeu de données, quelle que soit sa structure. C'est plus compliqué en terme de modèle et d'IHM, mais l'appli est bien meilleure d'un point de vue fonctionnel.

Contraintes techniques

Dans cette exercice, vous êtes en mode « démonstrateur » ou « Proof Of Concept ». Il s'agit de valider un ou plusieurs modèles et de démontrer les différentes fonctionnalités. A l'issue de ce travail, une industrialisation sera nécessaire. Pour l'instant donc, on ne sépare pas fonctionnellement les aspects « backend » et « frontend ». Vous verrez ça l'année prochaine, nous privilégions dans cette saé une approche « intégrée ».

- votre application s'appuie sur une base de données relationnelle, **interrogée avec SQL**, qui implémente un modèle que vous aurez pris soin de bien définir pour qu'il puisse couvrir les fonctionnalités énoncées.
- une IHM (Interface Homme Machine) permettant à l'utilisateur d'assurer le suivi de son jeu de données et naviguer parmi les trois fonctionnalités énoncées.
- Vous utiliserez le langage de programmation **Python**
- Vous utiliserez une base de données en mode « fichier » avec le module standard de Python **sqlite3** (https://docs.python.org/3/library/sqlite3.html). De cette manière l'application est auto-portante et peut être facilement déployée.
- Pour l'IHM, utilisez **streamlit**, qui permet de prototyper rapidement des IHM dans le navigateur (possibilité d'utiliser gradio, solara ou autres, à discuter avec le professeur).
- Pour assurer le traitement des données (entre la base SQL et l'IHM), vous représenterez les informations avec **Pandas** (https://pandas.pydata.org/).

Les jeux de données sur lesquels travailler
Dans l'archive zip, on donne trois fichiers jsonl correspondant à trois jeux de données :

1. python-codes-25k.jsonl : un ensemble de 25000 entrées concernant des codes Python (https://huggingface.co/datasets/flytech/python-codes-25k/blob/main/python-codes-25k.jsonl)

2. HuggingFaceH4/CodeAlpaca_20K : 20 000 entrées de code (java, sql, bash etc.)

3. jpacifico/French-Alpaca-dataset-Instruct-110K : plus de 1000000 lignes d'ensemble instruction/réponse en français

Calendrier :
Lancement le lundi 6 mai 2024
Fin le 13 juin 2024. Soutenances prévues le matin.

Livrables :
Un ou plusieurs programmes Python commentés
Un README.md, au format markdown, qui présente la procédure d'installation/de déploiement, et toute information permettant de bien utiliser l'application.

Organisation du travail :
Travail par groupe de 2 à 4 étudiants.
Séances encadrées « TP SAé » dans OGE. Séance en autonomie « Projets SAé » dans l'emploi du temps.
Utilisation privilégiée de github en mode privé.

Remarque : vous pouvez vous aider de tout outil vous permettant d'améliorer votre productivité, aussi bien pour la génération de code que pour les commentaires ou la documentation (chatgpt, copilot etc.).

Annexe1 - Rappel du référentiel (point 2.2.4 page 87)

2.2.4. SAÉ 2.04 : Exploitation d'une base de données

Compétence ciblée :
– Concevoir, gérer, administrer et exploiter les données de l'entreprise et mettre à disposition toutes les informations pour un bon pilotage de l'entreprise

Objectifs et problématique professionnelle :
La problématique professionnelle est de mettre des données dans une base de données et de les exploiter. Cette SAÉ permet une première approche complète des aspects de conception, implémentation, administration et exploitation d'une base de données.

Descriptif générique :
En partant d'un cahier des charges, il faut réaliser et étudier une base de données. À partir d'un jeu d'essais, il doit être proposé une visualisation des informations permettant d'apporter une analyse à l'entreprise.

Les livrables attendus généralement dans le monde professionnel sont :
– Étude des données et visualisation des informations
– Modèle de données
– Présentation orale des résultats en anglais
– Script de création de base de données

Apprentissages critiques :
– AC14.01 | Mettre à jour et interroger une base de données relationnelle (en requêtes directes ou à travers une application)
– AC14.02 | Visualiser des données
– AC14.03 | Concevoir une base de données relationnelle à partir d'un cahier des charges

Ressources mobilisées et combinées :
– R2.06 | Exploitation d'une base de données
– R2.08 | Outils numériques pour les statistiques descriptives
– R2.10 | Introduction à la gestion des systèmes d'information
– R2.12 | Anglais

Annexe 2 – Exécuter une app streamlit/pandas/datasets dans un container docker

Dans un terminal, depuis le dossier de l'application de test qui est fournie :

```
docker build -t tst_app --progress=plain .

docker run -p 8501:8501 tst_app
```

Annexe 3 – Exécuter une app streamlit/pandas/datasets en local

Dans un terminal, depuis le dossier de l'application :

```
pip install streamlit datasets pandas
```

```
streamlit run app.py
```