Projet : Fouille d'opinions dans les commentaires de clients

Cours de fouille de textes
Master 2 MIASHS/SSD - S. Aït-Mokhtar

La date limite pour rendre le projet est le 09 janvier 2026.
Il est possible de travailler seul ou au maximum à deux sur le projet.

1. Introduction

Le thème du projet est la fouille d'opinions dans les textes. L'objectif est l'implémentation d'un outil qui classifie les opinions globales des auteurs d'avis sur des restaurants selon 3 dimensions ou aspects prédéfinis :

1. Prix : concerne les prix des plats et boissons
2. Cuisine : concerne la qualité et la quantité de la nourriture et des boissons proposées.
3. Service : concerne la qualité et l'efficacité du service et de l'accueil.

Pour chaque avis et chaque aspect, l'outil doit prédire une des 4 valeurs (classes) possibles pour une opinion globale sur l'aspect, à savoir :

1. Positive : lorsque l'avis contient une ou plusieurs opinions positives sur l'aspect en question, et aucune opinion négative sur cet aspect.
2. Négative : lorsque l'avis contient seulement une ou plusieurs opinions négatives sur l'aspect en question et aucune opinion positive sur cet aspect.
3. Neutre : lorsque l'avis exprime au moins une opinion positive et une autre négative sur l'aspect en question.
4. NE (qui veut dire: Non Exprimée) : lorsque l'avis ne contient aucune opinion exprimée sur l'aspect en question.

2. Evaluation du rendu

La note globale our le cours Fouille de textes sera calculée en fonction des éléments suivants:

- L'exactitude (accuracy) moyenne sur les 3 aspects (sur 12 points): moyenne des 3 mesures d'exactitude (sur les 3 aspects). L'exactitude sur un aspect est le pourcentage d'avis pour lesquels les valeurs d'opinion prédites pour cet aspect sont correctes.
- Son efficacité de calcul (4 points): vitesse du processus d'entraînement (si entraînement de modèle il y a) et du processus d'inférence (prédiction des opinions sur la partie test).
- La qualité de la description de ce qui a été réalisé dans le fichier README.txt (sur 4 points)

IMPORTANT: Des points de pénalités seront appliqués sur la note finale si :

- Le projet est rendu en retard (-1 point par jour de retard)
- Si le programme ne fonctionne pas ou produit une erreur pendant l'exécution (au minimum -5 points)
- L'utilisation de librairies Python qui ne sont pas dans la liste des librairies autorisées, ou d'un identifiant de LLM qui n'est pas dans la liste des identifiants LLM autorisés (-10 pts)
- Si les conditions concernant le rendu ne sont pas respectées (nom de fichier, format, structure, etc. voir les sections ci-dessous)

3. Rendu du projet

Merci d'envoyer votre rendu de projet à l'adresse sa2015g@gmail.com en pièce jointe. Il doit satisfaire les conditions suivantes :

1. Le rendu doit être sous la forme d'un seul fichier compressé au format zip (autres formats non acceptés).
2. Le nom de ce fichier compressé doit être composé du nom ou des noms de famille des auteurs, dans ce format: `Nom1_Nom2.zip` (deux auteurs) ou `Nom.zip` (un seul auteur)
3. La taille du fichier compressé ne doit pas dépasser 2 Mo.

Par ailleurs, lorsque le fichier du rendu est décompressé, le répertoire racine (dont le nom est formé des noms des auteurs, comme expliqué ci-dessus) doit contenir directement les éléments suivants, et seulement ces éléments (pour plus de détails, consulter la section 6) :

Elément	Description
README.txt	Un fichier texte pur qui doit contenir les informations suivantes : 1. Le(s) nom(s) complet(s) de ou des auteur(s) du rendu (max=2 auteurs) 2. Un ou deux paragraphes décrivant le classifieur (type de représentation, type d'architecture, hyper-paramètres, ressources éventuellement utilisées, etc.) 3. L'exactitude moyenne que vous obtenez sur les données de dev.
src	Sous-répertoire contenant TOUS les fichiers nécessaires à l'exécution du projet avec la commande : `python run_project.py`
data	(facultatif): le répertoire data initialement fourni, sans aucune modification de son contenu

Supprimez donc tout autre fichier ou répertoire automatiquement généré par les outils de développement utilisés (checkpoints de modèle, logs, répertoire .idea de PyCharm, etc.) avant de compresser le répertoire et de l'envoyer par mail.

⚠ Avant de rendre le project, vérifier qu'en allant dans le répertoire `src` et en tapant la commande:

```
python runprojects.py
```

le programme s'exécute correctement.

4. Python et librairies autorisées

La version de Python requise pour le projet est 3.12.x

En dehors des librairies standard qui accompagnent automatiquement Python 3.12.x , la liste des librairies Python autorisées est la suivante:

1. pytorch 2.8.x
2. transformers 4.56.x
3. tokenizers 0.22.x
4. datasets 4.0.x (la librairie de HuggingFace)
5. openai 1.107.x (la librairie python openai)
6. lightning 2.5.x
7. pandas 2.3.x
8. numpy 2.1.x
9. jinja2 3.1.x
10. pyrallis 0.3.1
11. sentencepiece 0.2.0

L'utilisation de toute autre librairie ou de versions différentes pourrait faire échouer l'exécution de votre programme.

5. LLM autorisés

La liste des LLM qu'il est possible d'utiliser est la suivante :

- gemma2:2b
- llama3.2:1b
- qwen2.5:1.5b
- gemma3:1b
- qwen3:1.7b

Merci d'utiliser les identifiants tels qu'ils sont listés ici. Si vous avez déjà téléchargé un des modèles ci-dessus avec un autre identifiant, il faudra l'effacer (avec la commande `ollama rm identifiant_actuel`) et le re-télécharger avec la commande `ollama pull identifiant` (tel que `identifiant` est un des identifiants de LLM listés ci-dessus).

6. Comment procéder: étape par étape

6.1. Installation d'un environnement python et des librairies

Créez un nouvel environnement python réservé à ce projet, et appelez-le par exemple `ft` . Pour ce faire, le plus simple est d'installer `anaconda` ou `miniconda` si ce n'est pas encore fait, puis de créer un environnement conda nommé `ft` avec python 3.12 en faisant :

```
conda create -n ft python=3.12 ipython ipykernel
```

Ensuite, activez l'environnement et installez les librairies listées en section 4 (et seulement celles-ci).

Si vous comptez exploiter un LLM pour le projet, il faut également, comme nous l'avons vu ensemble lors des séances de TP:

- Installer le serveur Ollama
- Télécharger au moins un LLM, parmi ceux autorisés (voir section 5).

6.2. Téléchargement du code de base pour le rendu

A partir du sous-répertoire `projet` répertoire partagé du cours sur Google Drive, vous avez téléchargé le fichier compressé `ftproject.zip` .
En décompressant ce fichier, vous obtenez un répertoire racine `ftproject` qui contient :

1. Un sous-répertoire `src` contenant des fichiers de code sur lesquels votre contribution sera fondée.
2. Un sous-répertoire `data` contenant les données d'entraînement, de développement (ou validation) et de test. Ces fichiers de données ont le format tsv déjà vu lors des sessions TP. Ne pas toucher à ces fichiers, ils sont exploité par le code dans `src` .

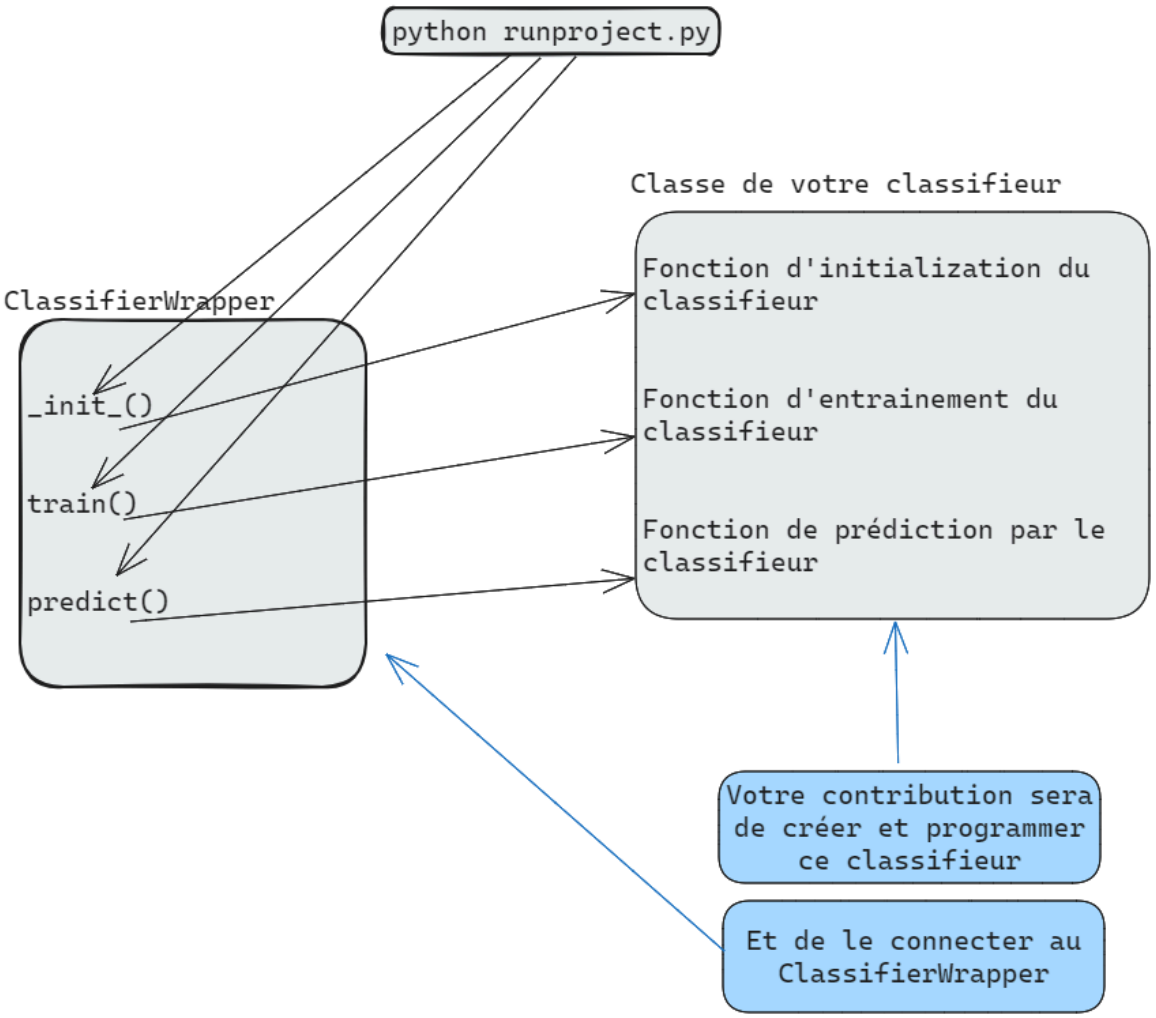
La première chose à faire est de renommer le répertoire racine pour qu'il soit formé du nom ou des noms des auteurs du rendus, de telle sorte que lorsque vous le compressez au format zip, le fichier compressé qui en résulte ait un nom compatible avec les directives énoncées en section 3 de ce document.

Il ne faudrait pas modifier la structure à l'intérieur de ce répertoire racine.

6.3. Développement du classifieur

Dans le répertoire `src` , les fichiers de code `config.py` et `runproject.py` ne doivent pas être modifié.
Le script principal est `runproject.py` et permet de tester l'exécution du rendu en créant un objet de type `ClassifierWrapper` , classe définie dans le fichier `classifier_wrapper.py` , et d'appeler deux de ses fonctions : `train()` pour entraîner le modèle de prédiction/classification des opinions si entraînement il y a, et `predict()` pour prédire les opinions et évaluer le classifieur en calculant l'exactitude sur les données de test.

Votre travail consiste à créer et programmer un classifieur dans un fichier de code séparé, puis de le connecter à `ClassifierWrapper` pour qu'il soit appelé indirectement par `runproject.py` , comme le montre le schéma ci-dessous:



Vous allez donc devoir modifier le contenu des fonctions `__init__()`, `train()` et `predict()` de `ClassifierWrapper` pour y connecter votre classifieur. Cependant, il ne faut pas modifier les signatures de ces trois fonctions (leur nom, arguments et type de données retourné ne doit pas changer).

La classe `ClassifierWrapper` possède également une variable `METHOD`. Sa valeur doit être `'LLM'` si vous optez pour une solution avec un LLM sans entraînement, et `'PLMFT'` (Pretained-Language Model Fine-Tuning) si vous optez pour une solution à base de PLM à entraîner sur la partie `train` des données distribuées.

En guise d'exemple, il y a un fichier `llm_classifier.py` qui définit une classe `LLMClassifier` qui implémente un classifieur à base de LLM, comme nous l'avons vu en TP.

Dans un terminal de commandes, après avoir activé votre environnement python pour le projet, et en allant dans le répertoire `src`, la commande suivante permet de tester et d'évaluer le classifieur :

```
python runproject.py
```

Vous pouvez specifier le nombre d'exemples des données de test qui sont utilisés pour effectuer l'évaluation : utile si vous souhaitez effectuer des test rapides sur une petite partie des données, pendant le développement de votre classifieur. Par exemple, pour effectuer le test sur seulement 40 exemples :

```
python runproject.py --n_test=40
```

De même, vous pouvez contrôler le nombre d'exemples d'entraînement et le nombre de runs à effectuer : ces deux options ont un sens seulement si vous développez un classifieur avec un modèle à entraîner (dans le cas contraire, ces options seront ignorées)

```
# Effetcuer 3 runs, chacun avec un entraînement du modèle sur 1000 exemples seulement
python runproject.py --n_train=1000 --n_runs=3
```

6.4. Important

1. Votre code doit fonctionner sans erreur, en allant dans le répertoire `src`, et en tapant la commande `python runproject.py` et ce sans modifier ni `runproject.py` ni `config.py`, ni les signatures des fonctions de la classe `ClassifierWrapper`.
2. Vous pouvez rajouter dans `src` d'autres fichiers de code python si nécessaires.
3. Vous n'avez pas besoin d'écrire du code python pour lire les données (du répertoire `data`) : cette lecture est déjà réalisée dans le script `runproject.py` et vous récupérez les données sous forme de liste à partir des arguments des fonctions `train()` et `predict()` de la classe `ClassifierWrapper`.

4. Relire la section 3 ci-dessus concernant le contenu du rendu et les contraintes à respecter avant d'envoyer le rendu.