Analyser les émotions d'un journal intime

Contexte du projet

Vous travaillez au sein d'une équipe data dans une ESN. Un client vous sollicite afin de developper un outil basé sur l'IA pour l'aider dans son travail.

Votre client est un coach de vie, il aide les personnes à se sentir bien dans leur quotidien. Il demande à ses patients d'écrire un journal intime. Chaque jour, ils doivent écrire comment ils se sentent.

Le coach, M. Zen avait l'habitude d'associer chaque entrée du journal intime à une émotion. Cependant, il y a quelque temps, un youtubeur a conseillé M. Zen à sa communauté. Depuis cette recommandation, il est totalement débordé par l'explosion de la demande. Il n'est plus en mesure de travailler comme avant et il aimerait automatiser le suivi du journal intime grâce à un outil numérique.

Votre objectif est double. Vous devez dans un premier temps développer un algorithme qui pourra associer chaque entrée du journal intime à une émotion et développer une application web qui servira d'intermédiaire entre le coach et ses patients.

Modélisation

Premièrement, vous devez développer 3 modèles de classification des émotions basées sur les packages suivants :

- Scikit-learn
- Keras
- TextHero et FastText
- *Bonus : Hugging Face (+Tensorflow)*

Tester et mesurer l'impact des divers méthodes de prétraitement : suppression des stop-words, lemmatisation, n-grams et différentes approches de vectorisation.

Pour entraîner vos modèles, vous devrez travailler dans un premier temps avec le jeu de données issue de Kaggle.

Ensuite, vous allez procéder à une augmentation du jeu de données d'entraînement afin tenter d'améliorer la qualité des prédictions. Pour cela vous utiliserez ce jeu de données issu de data.world. Attention, les classes sont légérement différentes, servez-vous de la roue des émotions pour harmoniser les datasets.

Mesurer l'impact de cette *data augmentation* (par exemple en testant les performances de vos modèles sur le jeu de données test de Kaggle avec et sans data augmentation).

Comparer les résultats obtenus dans vos 4 modélisations et sélectionner la plus performante afin de la déployer.

Application

Une fois la partie modélisation effectuée, vous allez développer un outil pour ce coach de vie, composé de 3 élèments :

- Une base de données afin de stocker les informations des personnes suivies par le coach.
- Une REST API (par exemple avec fastAPI) pour pouvoir interagir avec cette base de données.
- Une application web avec streamlit. Votre supérieur insiste pour que vous développiez cette application avec Streamlit et non avec Flask afin de vous focalisez sur la partie Machine Learning du projet.

Ce qu'un patient doit pouvoir faire:

- Ajouter un texte à la date du jour.
- Modifier un texte à la date du jour.
- Lire son texte à la date du jour ou à n'importe quelle date.

Ce que le coach doit pouvoir faire:

- Ajouter/supprimer/renommer un patient.
- Ajouter/supprimer/modifier certaines informations sur le patient.
- Obtenir la liste de tous ses patients et les informations stockées sur lui.
- Pour un certain patient à une certaine date obtenir le texte d'un patient, son sentiment majoritaire, sa roue des émotions (% de chaque sentiment)
- Pour un certain patient, un certain jour, une certaine semaine, un certain mois ou une certaine année: récupérer la roue des sentiments moyenne sur la période.
- Pour un certain jour, une certaine semaine, un certain mois, une certaine année: récupérer la roue des sentiments moyenne de l'ensemble de ses patients sur la période.

Conseil: utiliser le jeu de données test pour simuler les contenus passés.

Vous pouvez déployer votre application sur Heroku ou Microsoft Azure.

Bonus : déployer l'application sur le site de hugginface (gradio).