**SentimentGPT-** Génération de texte sensible aux émotions

**Objectifs du projet**

L'idée de ce projet était de créer un système capable de :

* détecter l’émotion dans un message écrit par un utilisateur,
* générer une réponse adaptée à cette émotion (ou à une émotion choisie),
* bref, rendre un chatbot un peu plus empathique et humain dans ses réponses.

**Ce qu’on a fait, étape par étape**

**1. Recherche / État de l’art**

Pour commencer, on a analysé et examiné les travaux déjà existants autour de la détection d’émotions en NLP. On s’est notamment intéressé à des modèles comme BERT pour la classification, et à des approches de génération de texte contrôlée avec GPT-2 ou T5.

On a aussi exploré plusieurs techniques comme :

* le **prompt tuning** (ajouter des mots ou instructions pour guider la réponse),
* le **prefix tuning** (influencer directement les couches du modèle),
* ou encore le **fine-tuning complet**, où on réentraîne le modèle avec des données spécifiques.

**2. Collecte et préparation des données**

On a utilisé plusieurs datasets annotés en émotions :

* **GoEmotions** (proposé par Google),
* **MELD** (utile pour les dialogues multi-locuteurs),
* **ISEAR** (sur des expériences personnelles et les émotions associées).

Une fois les données récupérées, on les a nettoyées, tokenisées, vectorisées, bref, on les a préparées pour l’entraînement du modèle.

**3. Détection d’émotions**

À partir de là, on a fine-tuné un modèle BERT sur ces données pour qu’il puisse reconnaître les émotions principales (joie, colère, tristesse, peur, surprise, dégoût, neutre).  
On a ensuite testé les performances avec des métriques classiques (accuracy, F1-score) et les résultats étaient globalement satisfaisants.

**4. Génération de texte émotionnelle**

Une fois la détection d’émotions en place, on a utilisé GPT-2 pour générer des réponses adaptées.

On a testé plusieurs façons d’intégrer l’émotion :

* ajouter un **token émotionnel** (comme [EMO\_JOIE]) au début du message,
* utiliser du **prompt engineering** pour influencer la sortie,
* ou entraîner le modèle directement sur des données annotées émotionnellement (petit fine-tuning).

**5. Évaluation**

On a évalué les réponses générées selon plusieurs critères :

* la **cohérence grammaticale** (automatiquement avec la perplexité + manuellement),
* la **pertinence par rapport au message de départ**,
* et surtout **l’adéquation entre la réponse et l’émotion attendue**.

On a utilisé quelques métriques automatiques (BLEU, ROUGE) mais aussi des évaluations à la main.

**6. Interface**

Enfin, on a développé une petite interface avec **Streamlit** :

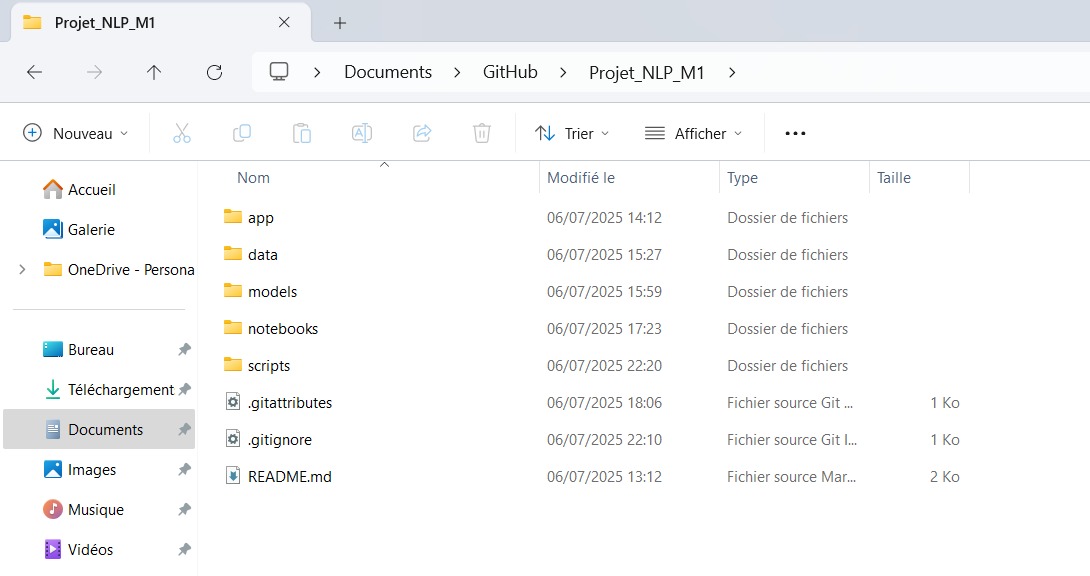
* l’utilisateur tape son message,
* l’émotion détectée s’affiche,
* une réponse est générée automatiquement, selon l’émotion détectée ou une autre choisie.

**Technos utilisées**

* Python, Pytorch, Transformers (HuggingFace)
* Pandas, Matplotlib
* Streamlit
* Environnement : Google Colab / Jupyter

**Résultats**

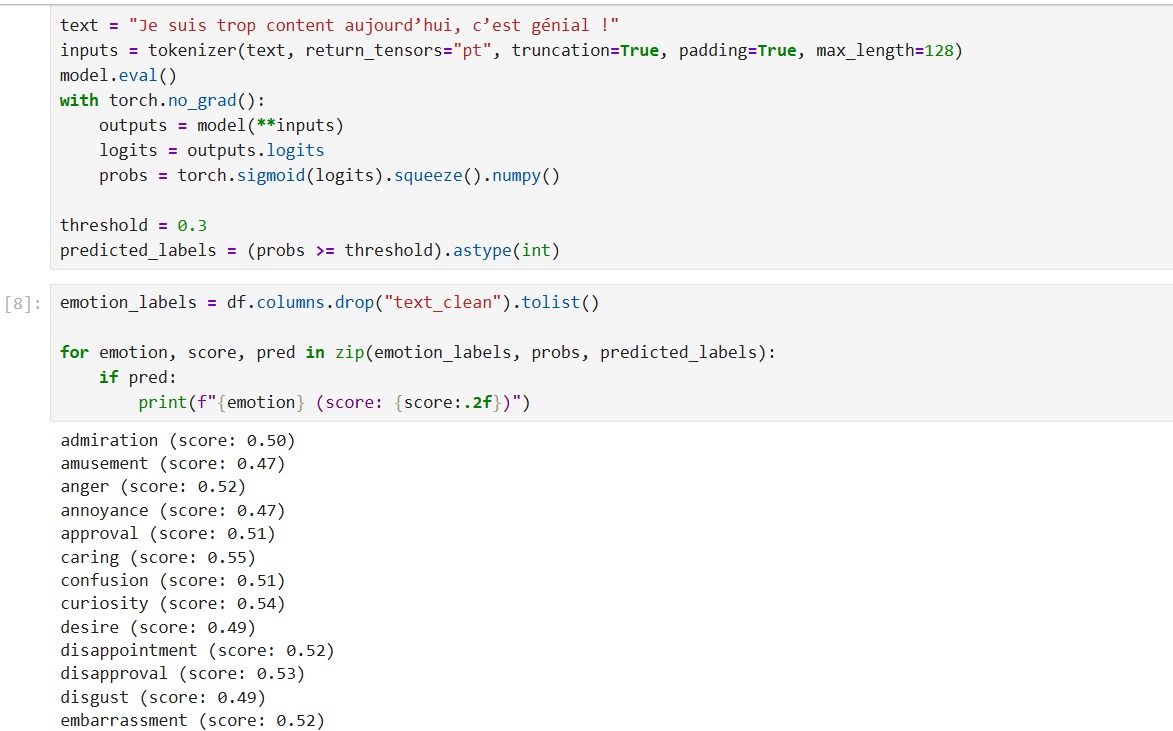
* La détection d’émotions fonctionne bien (précision autour de 80% sur GoEmotions).
* Les réponses générées sont globalement fluides et cohérentes avec les émotions.
* L’interface est fonctionnelle et simple à utiliser.



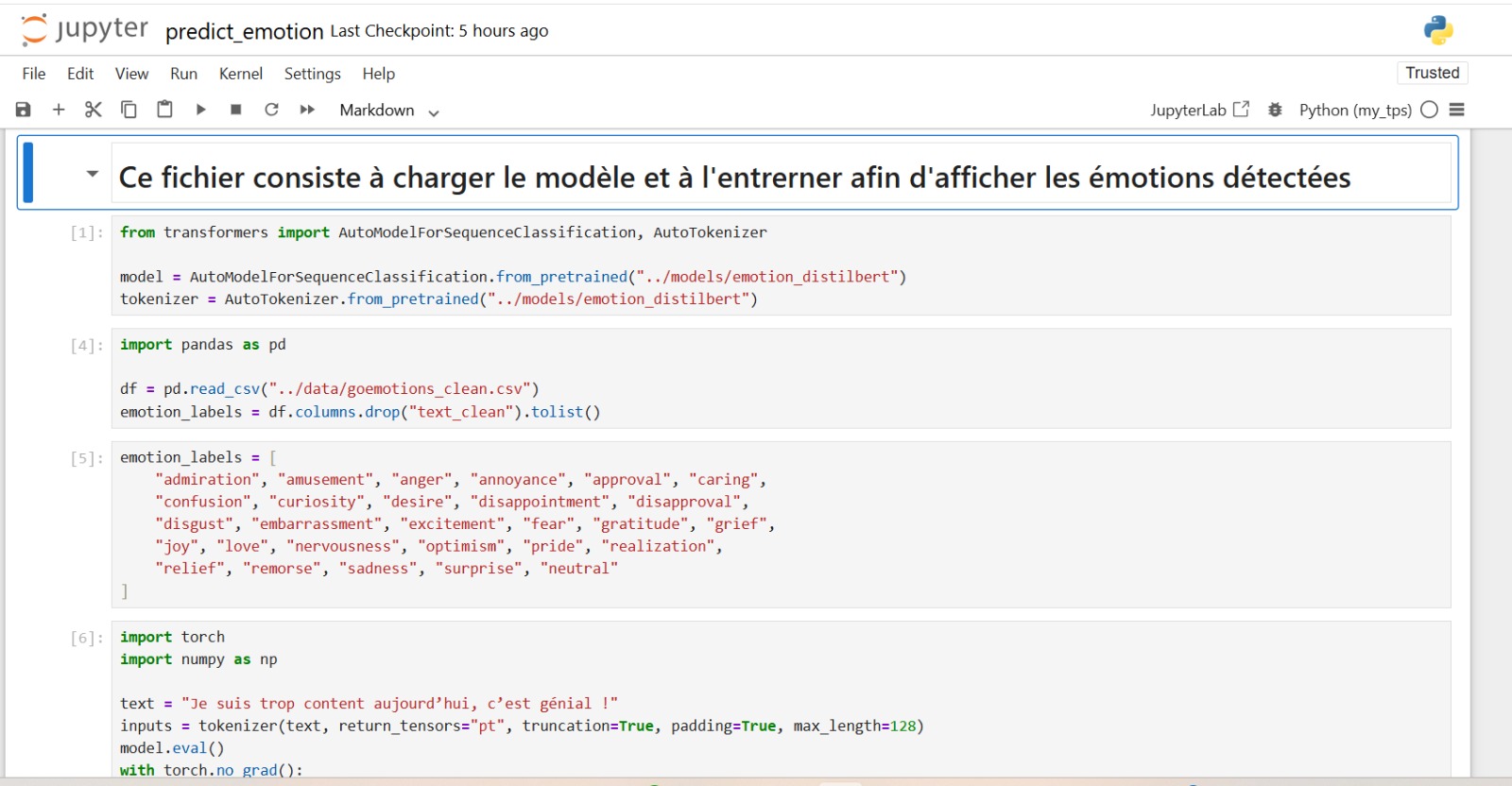
## Preprocessing



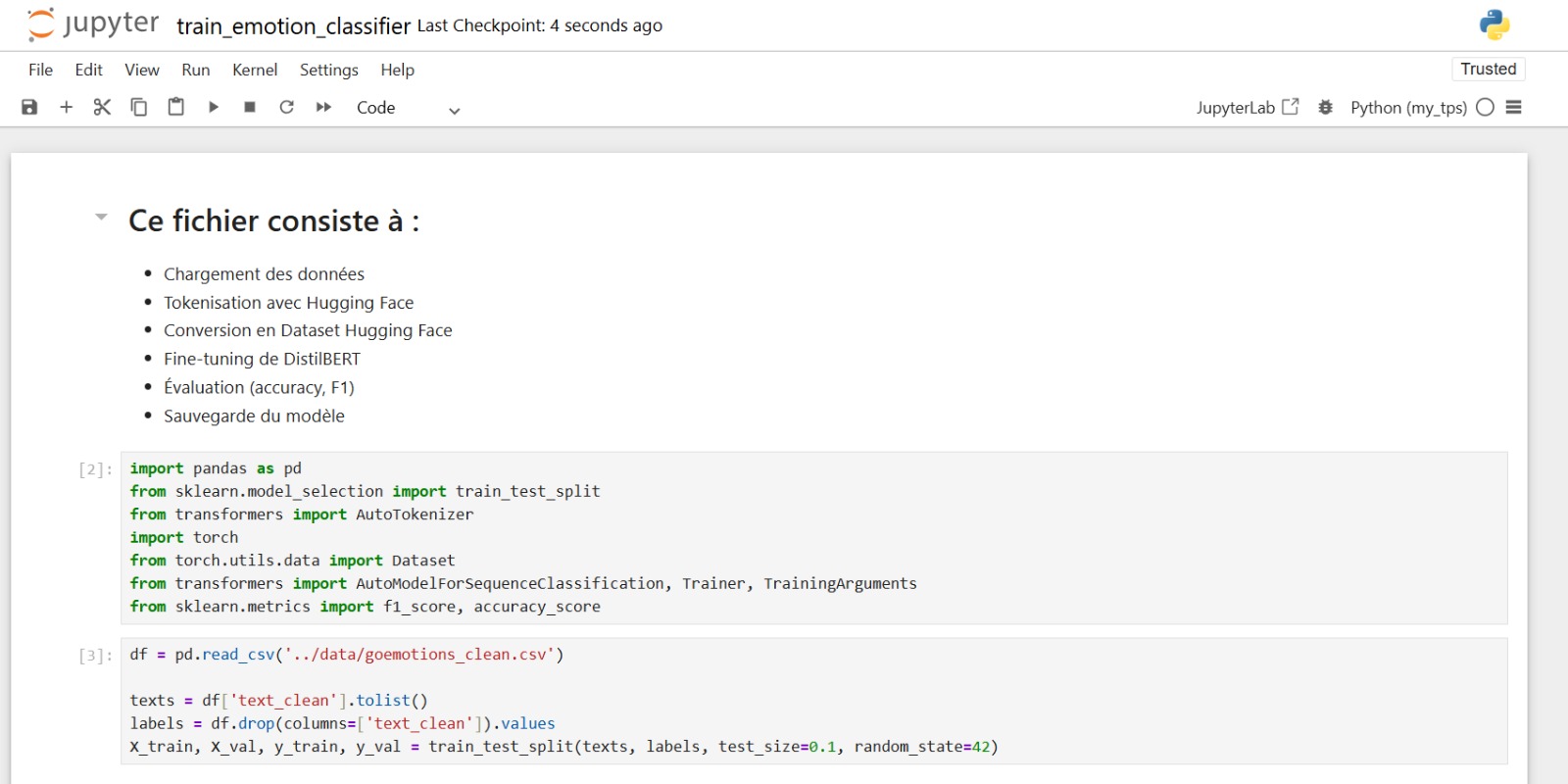
## Exploration visuelle



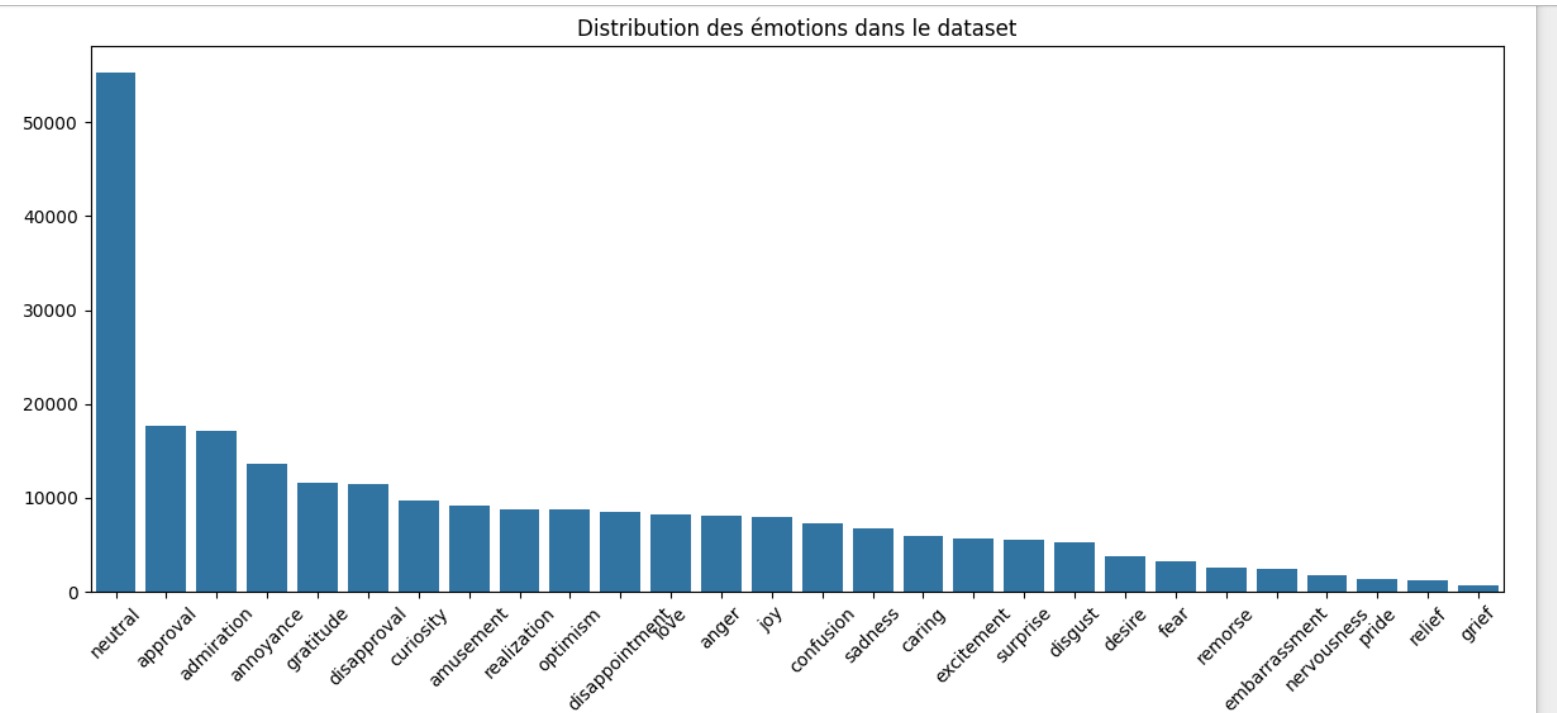
## Structure projet



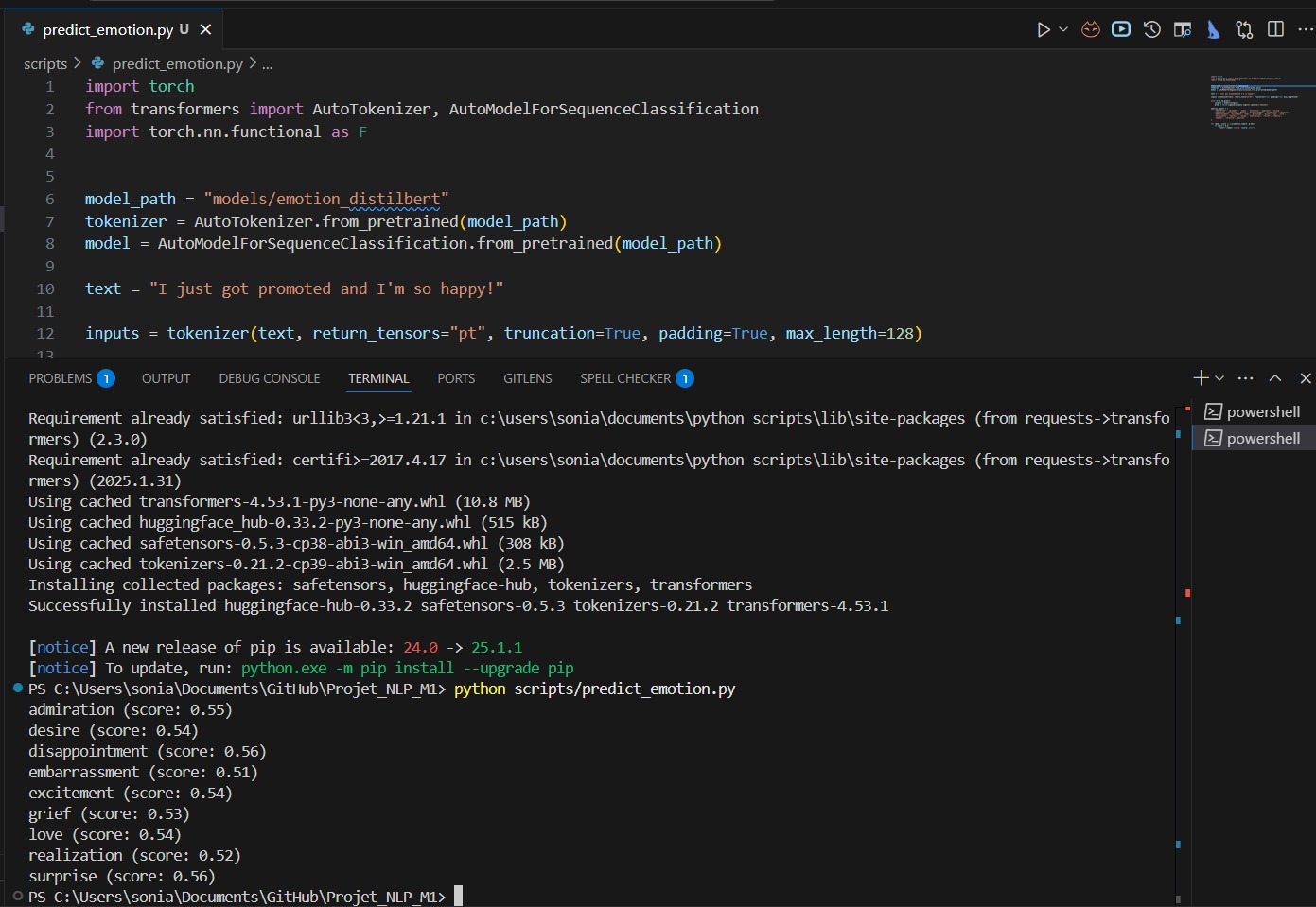
## Modèle d’entraînement



## Utilisation du modèle



## Résultats du modèle - 1



## Résultats du modèle - 2

**Conclusion**

Ce projet m’a permis de mettre en pratique plusieurs notions vues en cours : classification NLP, génération de texte avec GPT, et un peu d’interface utilisateur.  
C’était super intéressant de voir comment on peut combiner plusieurs briques (détection + génération) pour créer un outil qui peut rendre l’interaction homme-machine un peu plus “humaine”.