**TRAITEMENT ET DETERMINATION DES SENTIMENT**   
  
approche pour combiner **à la fois** la polarité du texte (via TextBlob) **et** la note score (1 à 5) dans une même **fonction de calcul du sentiment**. L’idée est de :

1. **Convertir** la note score en un **intervalle** semblable à la polarité de TextBlob (ex. -1 à +1).
2. **Combiner** la polarité textuelle et le “score transformé” avec des **coefficients** de pondération.
3. **Classer** le résultat final (positive, negative, neutre, etc.).

Cela peut donner un résultat plus **réaliste** que la polarité textuelle seule, en tenant compte du fait qu’un utilisateur qui met la note 1 a de fortes chances d’avoir un sentiment négatif, même si le texte est court ou ironique.

**1) Transformer la note (1–5) en intervalle (-1 … +1)**

Par exemple :

* Note = 1 ⇒ Score transformé = -1.0
* Note = 5 ⇒ Score transformé = +1.0
* Note = 3 ⇒ Score transformé = 0.0  
  Et ainsi de suite.

La formule de transformation possible :

score\_scaled=(score−3)2\text{score\\_scaled} = \frac{(score - 3)}{2}

* score = 1 ⇒ (1−3)/2=−1.0(1 - 3)/2 = -1.0
* score = 5 ⇒ (5−3)/2=+1.0(5 - 3)/2 = +1.0
* score = 3 ⇒ (3−3)/2=0(3 - 3)/2 = 0

*(On peut bien sûr ajuster cette échelle si on souhaite par exemple un intervalle 0–1.)*

**2) Combiner polarité et note**

Disons que la **polarité** TextBlob (appelons-la text\_polarity) est dans [−1,+1][-1, +1].  
On peut faire une **moyenne pondérée**, du style :

combined=α×text\_polarity+β×score\_scaled\text{combined} = \alpha \times \text{text\\_polarity} + \beta \times \text{score\\_scaled}

avec α+β=1\alpha + \beta = 1 (ou pas forcément, c’est à toi de choisir les pondérations).

Exemple :

* α=0.7\alpha = 0.7 (70% de poids sur le contenu textuel)
* β=0.3\beta = 0.3 (30% de poids sur la note numérique)

De cette manière, si la personne a laissé une note de 1, mais le texte est neutre, le **résultat final** aura quand même tendance à être négatif à cause du score. Et inversement, si la personne a mis 5 étoiles mais que le texte semble plutôt neutre, le résultat final sera plutôt positif.

**3) Classification finale**

On peut garder un **seuil** pour dire :

* combined >= 0 ⇒ “positive”
* combined < 0 ⇒ “negative”

ou bien définir un **troisième** label “neutral” selon un intervalle (ex. [−0.1,+0.1][-0.1, +0.1]).

**Exemples de code**

Ci-dessous, un **script complet** qui :

1. **Lit** un CSV contenant a minima les colonnes content (texte) et score (1–5).
2. Calcule la polarité du texte via TextBlob (et optionnellement, gère les émojis).
3. Transforme la note en [−1,+1][-1, +1].
4. Calcule un **score combiné**.
5. Classe le sentiment final (“positive” / “negative” ou plus).
6. Enregistre le résultat dans un nouveau CSV.

import pandas as pd

from textblob import TextBlob

import emoji

def generate\_sentiment\_with\_score\_csv(

input\_csv: str,

output\_csv: str = "uber\_data\_with\_combined\_sentiment.csv",

alpha: float = 0.7

) -> pd.DataFrame:

"""

Lit le CSV 'input\_csv', génère une colonne 'sentiment'

(basée sur polarité texte + score utilisateur),

et enregistre le résultat dans 'output\_csv'.

Paramètres

----------

input\_csv : str

Chemin vers le fichier CSV d'entrée.

output\_csv : str

Chemin vers le fichier CSV de sortie (avec sentiment).

alpha : float

Pondération pour la polarité du texte.

(1 - alpha) sera la pondération pour le score transformé.

Retour

------

pd.DataFrame

Le DataFrame résultant, contenant la nouvelle colonne 'sentiment'.

"""

# Lecture du CSV

df = pd.read\_csv(input\_csv)

# 1) Fonction pour calculer la polarité via TextBlob

def get\_text\_polarity(text: str) -> float:

if not isinstance(text, str):

return 0.0

# Optionnel : demojize

text\_no\_emoji = emoji.demojize(text)

# Calcul de la polarité

return TextBlob(text\_no\_emoji).sentiment.polarity

# 2) Fonction pour transformer un score (1..5) en [-1..+1]

def scale\_score(score: float) -> float:

# Ex: 1 => -1, 5 => +1, 3 => 0

return (score - 3) / 2

# 3) Combiner polarité texte + note

def combined\_sentiment(text: str, score: float) -> float:

pol\_text = get\_text\_polarity(text)

pol\_score = scale\_score(score)

# combined = alpha \* polarité\_texte + (1-alpha) \* polarité\_score

return alpha \* pol\_text + (1 - alpha) \* pol\_score

# 4) Classer en label final

def classify\_combined(value: float) -> str:

# Seuil simple : >= 0 => positive, < 0 => negative

return "positive" if value >= 0 else "negative"

# Vérifions que 'content' et 'score' existent dans df

if 'content' not in df.columns or 'score' not in df.columns:

raise ValueError("Le CSV doit contenir au moins les colonnes 'content' et 'score'.")

# Calcul du score combiné

df['combined\_score'] = df.apply(

lambda row: combined\_sentiment(row['content'], row['score']),

axis=1

)

# Classification finale

df['sentiment'] = df['combined\_score'].apply(classify\_combined)

# Sauvegarde

df.to\_csv(output\_csv, index=False)

return df

if \_\_name\_\_ == "\_\_main\_\_":

# Exemple d'appel direct

df\_sent = generate\_sentiment\_with\_score\_csv(

input\_csv="Assets/Datas/archive\_uber/uber\_data\_cleaned.csv",

output\_csv="Assets/Datas/archive\_uber/uber\_data\_with\_combined\_sentiment.csv",

alpha=0.7 # 70% polarité texte, 30% note

)

print("Sentiment généré avec prise en compte du score utilisateur.")

**Explications**

1. **get\_text\_polarity** :
   * Convertit d’abord les émojis en codes :smiley: si tu le souhaites (via emoji.demojize).
   * Calcule la polarité avec TextBlob sur [−1,+1][-1, +1].
2. **scale\_score** :
   * Transforme la note (1..5) en [−2/2..+2/2][-2/2..+2/2] = [−1..+1][-1..+1].
   * (Note de 3 => 0, 1 => -1, 5 => +1.)
3. **combined\_sentiment** :
   * Fait la moyenne pondérée \alpha \cdot \text{text\_polarity} + (1-\alpha) \cdot \text{score\_scaled}.
   * **alpha** est un paramètre qu’on peut ajuster (ex. 0.7 = plus de poids au texte ; 0.3 = note user).
   * Si tu veux plus de poids au score, mets alpha plus faible (ex. 0.4).
4. **classify\_combined** :
   * Simple logique binaire (>= 0 => “positive”, < 0 => “negative”).
   * Tu peux introduire un label “neutral” si la valeur est entre -0.1 et 0.1, par exemple.

**4) Ajustements et améliorations possibles**

* **Trois labels** : “positive”, “negative”, “neutral”. Par exemple,
* def classify\_combined(value: float) -> str:
* if value > 0.1:
* return "positive"
* elif value < -0.1:
* return "negative"
* else:
* return "neutral"
* **Pondération dynamique** : si la note est extrême (1 ou 5), on peut accentuer le poids du score ; si la note est moyenne (3), on laisse plus de place à la polarité textuelle, etc.
* **Échelle différente** : on pourrait transformer le score en 0..1 au lieu de -1..+1.
* **Enrichir la pipeline** d’analyse textuelle (retrait de stopwords, lemmatisation, etc.) pour améliorer la pertinence de la polarité.

**Conclusion**

Avec ce **système combiné**, tu :

1. **Exploites** le score (1..5) fourni par l’utilisateur,
2. **Mélanges** cette information à la **polarité du texte** (TextBlob) pour **affiner** la conclusion,
3. **Obtiens** un fichier final où chaque ligne dispose d’un combined\_score et d’un label sentiment tenant compte **et** du rating **et** du contenu textuel.