



Restaurant

REVIEWS

RAPSODE Sophia - REN Roseline



SUMMARY

1. Presentation du projet & Contexte
2. Exploration/Transformation/feature engineering
3. Modélisation et Interprétation des résultats
4. Application



But & Contexte

- **Contexte:** Yelp est une plateforme qui permet aux utilisateurs de rechercher et de découvrir des avis sur des restaurants. Les utilisateurs peuvent laisser des critiques, noter les établissements et partager leurs expériences sur Yelp.

- **But:** Faciliter l'analyse des avis pour mieux comprendre l'expérience client.

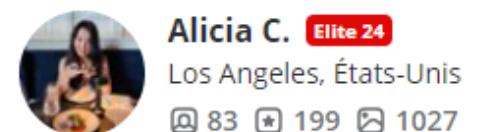
- **Techniques utilisées:** Correction de texte, analyse de sentiments, génération de résumé, ABSA et dialo-GPT



1. Le Comptoir de la Gastronomie
 4.5 (1.3k review)
Français €€ • Châtelet/Les Halles
Open until Minuit



2. Bistro des Augustins
 4.4 (486 reviews)
Bistrot Bars à vins €€ • Saint-Michel/Odéon
Open until 02:00



25 oct. 2023
3 photos

This is one of my favorite restaurants in Paris. I come here every time I'm there and it's one of the places I recommend to all my friends if they are planning a trip. Highly suggest getting reservations ahead of time if you can.

Service is always top-notch, food is seasoned well, cooked perfectly, and prices are very reasonable. My go to starter is the duck foie gras carpaccio. The balance of sweet and salty from the honey sauce and sea salt is the best combination with the silky foie gras on toasted bread. You really can't go wrong with any dish or drinks here as I have not had a bad experience so far.

Just a heads up, the dining tables are small and set close to each other so there is not much privacy while dining but on the bright side, you do get to meet and chat with others dining near you.



Présentation des données

- Restaurants & avis: on récupère depuis l'API Yelp 500 restaurants situés à Paris et leurS avis**
- API limitation: 3 avis par restaurants et 500 requêtes par jours**

business_id	review_id	text	rating	time_created
-0iLH7iQNYtoURciDpJf6w	sxEfkJ89kyF-wMDUI2ZnWw	Based on the menu presented, one could write a...	5	2023-12-29 23:14:16
-0iLH7iQNYtoURciDpJf6w	n3IGqJHkjPc6SVDH0VbSNw	I like this place but it does feel a little bi...	4	2024-01-03 09:01:14
-0iLH7iQNYtoURciDpJf6w	3MYKaD-tDrUVhRgDh9G4dA	If you love French Onion Soup, this is for you...	5	2023-12-14 13:59:59
IU9_wVOGBKjfqTTpAXpKcQ	PJuWhEzKFz3ipwhOcWMMBA	Came here with my daughter, son inlaw his mom...	5	2023-09-22 09:36:00
IU9_wVOGBKjfqTTpAXpKcQ	T2k7VQB-TFkzkaEXRDfwYw	This tiny eatery along the Seine on the left b...	5	2024-01-08 16:55:09

id	alias	name	image_url	is_closed	url	review_count	categories	rating	coordinates	transactions	price	location	phone	display_phone
-0iLH7iQNYtoURciDpJf6w	le-comptoir-de-la-gastronomie-paris	Le Comptoir de la Gastronomie	https://s3-media3.fl.yelpcdn.com/bphoto/xT4YkC... media3.fl.yelpcdn.com/bphoto/xT4YkC...	False	https://www.yelp.com/biz/le-comptoir-de-la-gas...	1304	[{"alias": "french", "title": "French"}]	4.5	{"latitude": 48.8645157999652, "longitude": 2...	[]	€€	{"address1": "34 rue Montmartre", "address2": ...}	+33 1 42 33 31 32	+33 1 42 33 31 32
IU9_wVOGBKjfqTTpAXpKcQ	bistro-des-augustins-paris	Bistro des Augustins	https://s3-media2.fl.yelpcdn.com/bphoto/ctHDHM... media2.fl.yelpcdn.com/bphoto/ctHDHM...	False	https://www.yelp.com/biz/bistro-des-augustins-...	486	[{"alias": "bistros", "title": "Bistros"}, {"a...	4.5	{"latitude": 48.854754, "longitude": 2.342119}	[]	€€	{"address1": "39 quai des Grands Augustins", "...}	+33 1 43 54 04 41	+33 1 43 54 04 41
cEjF41ZQB8-SST8cd3EsEw	l-avant-comptoir-paris-3	L'Avant Comptoir	https://s3-media2.fl.yelpcdn.com/bphoto/V38oU4... media2.fl.yelpcdn.com/bphoto/V38oU4...	False	https://www.yelp.com/biz/l-avant-comptoir-pari...	657	[{"alias": "tapas", "title": "Tapas Bars"}, {"...	4.5	{"latitude": 48.85202, "longitude": 2.3388}	[]	€€	{"address1": "3 carrefour de l'Odéon", "addres...}	+33 1 42 38 47 55	+33 1 42 38 47 55

Structure du projet

- Utilisation de l'API:
NLP_Project2.ipynb
- Preprocessing:
NLP_Project2_preprocessing.ipynb
- Analyse de données:
NLP_Project2_analysis.ipynb
- Modélisation:
NLP_Project2_model.ipynb
- Application: App.py
- Dataset: restaurants.csv et
review_final.csv

Projet2_NLP Private

Unwatch 1

main Go to file + <> Code

roselineren Merge ... 517a87c · 40 minutes ago 14 Commits

File	Action	Last Commit
tensorboard-logs	update	42 minutes ago
.gitattributes	Initial commit	3 weeks ago
.gitignore	Initial commit	3 weeks ago
NLP_Project2.ipynb	update	yesterday
NLP_Project2_pre...	update	yesterday
NLP_Project_2_An...	update	17 hours ago

Lien Github

Problèmes

- Concernant l'expérience client dans un restaurant il y a plusieurs éléments qui contribuent à une bonne expérience pour le client: le goût des plats, la présentation, l'ambiance du lieu, la situation géographique le service, etc
- L'analyse de sentiments classique ne permet pas de faire la nuance entre tous ces éléments dans un avis. De même pour les notes qui portent sur l'ensemble de ces points.

Solution: Fournir un outil qui permet de relever le sentiment accordé à chaque critère évoqué dans l'avis et construire une synthèse de l'ensemble des avis sur un restaurant donné.

Exemple: En prenant un avis nuancé, un modèle d'analyse de sentiment classique (distilBert), le résultat renvoyé est souvent neutre bien que plusieurs critères entre en compte dans la décision du client.

“The ambiance was lovely as well as the service, but the food was not that good.”

```
10 # Exemple d'utilisation
11 example_review = "The food was really good but the service was lacking."
12 result = sentiment_analysis(example_review)
13
14 # Afficher le sentiment au lieu du label
15 for res in result:
16     sentiment = convert_label_to_sentiment(res)
17     print(f"Sentiment: {sentiment}, Score: {res['score']}")
```

```
10 # Exemple d'utilisation
11 example_review = "The ambiance was lovely as well as the service , but the food was not that good."
12 result = sentiment_analysis(example_review)
13
14 # Afficher le sentiment au lieu du label
15 for res in result:
16     sentiment = convert_label_to_sentiment(res)
17     print(f"Sentiment: {sentiment}, Score: {res['score']}")
```

✓ 0.0s

Sentiment: neutre, Score: 0.7484728693962097

“The food was really good but the service was lacking.”

Présentation des approches:

Approches: Comparaison des techniques d'analyse de sentiments pour déterminer quelle méthode serait la plus utile à l'étude des avis.

→ Pour cela on ajoutera des labels en considérant que la note accordée par le client reflète son sentiment à propos de son expériences vécue sur place.

Résultats: Analyse permettant d'obtenir le sentiment de façon plus précises sur les opinions des utilisateurs concernant des éléments spécifiques.

2

Exploration/Transformation/feature engineering

1. Nettoyage des données
2. Exploration simple
3. Embedding pour Identifier les mots similaires
4. Features ajoutés

2.1 Nettoyage des données

Suppression des doublons et vérification des valeurs nulles

```
# Supprimer les doublons basés sur 'review_id' s'ils existent
review_df.drop_duplicates(subset='review_id', inplace=True)
```

```
review_df.isna().sum()

business_id      0
review_id        0
text              0
rating            0
time_created     0
user_id           0
dtype: int64
```

Sur les deux datasets
qu'on a chargé :
restaurant_df et
review_df

```
print(restaurant_df.shape)

(500, 17)

len(restaurant_df['reviews'])!=0

True

restaurant_df=restaurant_df[(restaurant_df['reviews'].apply(lambda x: x != []))]

restaurant_df.shape

(486, 17)

np.size(restaurant_df['id'].unique())
```

2.1 Nettoyage des données

Correction des fautes de grammaire et d'orthographes, puis traduction

```
# Load model directly
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM

tokenizer = AutoTokenizer.from_pretrained("vennify/t5-base-grammar-correction")
model = AutoModelForSeq2SeqLM.from_pretrained("vennify/t5-base-grammar-correction")
```

```
from transformers import pipeline

fix_spelling = pipeline("text2text-generation",model="oliverguhr/spelling-correction-english-base")
```

```
from langdetect import detect
from transformers import MarianMTModel, MarianTokenizer
```

Utilisation des
transformers pour les
fautes de grammaire et
pour la traduction

2.1 Nettoyage des données

Correction des fautes de grammaire et traduction

review_df[['text', 'corrected_text', 'translated_text']].head(10)

✓ 0.1s Python

	text	corrected_text	translated_text
0	Based on the menu presented, one could write a...	Based on the menu presented, one could write a...	Based on the menu presented, one could write a...
1	I like this place but it does feel a little bi...	I like this place but it does feel a little bi...	I like this place but it does feel a little bi...
2	If you love French Onlon Soup, this is for you...	If you love French Onlon Soup, this is for you...	If you love French Onlon Soup, this is for you...
3	Came here with my daughter, son inlaw his mom...	Came here with my daughter, son in law, his mo...	Came here with my daughter, son inlaw his mom...
4	This tiny eatery along the Seine on the left b...	This tiny eatery along the Seine on the left b...	This tiny eatery along the Seine on the left b...
5	The food was AMAZING, I had the vegetarian veg...	The food was AMAZING, I had the vegetarian veg...	The food was AMAZING, I had the vegetarian veg...
6	I'm staying in the area and was sitting at a c...	I'm staying in the area and was sitting at a c...	I'm staying in the area and was sitting at a c...
7	I can't help but share my incredible experienc...	I can't help but share my incredible experienc...	I can't help but share my incredible experienc...
8	This is an amazing bar style restaurant. The s...	This is an amazing bar style restaurant. The s...	This is an amazing bar style restaurant. The s...
9	I understand that everyone wants to see food, ...	I understand that everyone wants to see food, ...	I understand that everyone wants to see food, ...

2.1 Nettoyage des données

Preprocessing : voir les commentaires du code

```
def preprocess_text(text):

    contractions = {
        "n't": " not",
        "'ve": "",
        "'s": "",
        "'m": " am",
        "'ll": " will",
        "'d": " would",
        "'re": " are",
        "'t": " not"
    }
    for contraction, replacement in contractions.items():
        text = re.sub(contraction, replacement, text.lower())

    # Suppression des caractères non alphanumériques et mise en minuscules
    text = re.sub(r"[\^\\w\\s]", "", text.lower())

    # Tokenisation
    tokens = word_tokenize(text)

    # Filtrage des stopwords
    tokens = [word for word in tokens if word not in stop_words]

    # Obtention des tags POS
    nltk_pos_tags = pos_tag(tokens)
```

```
# Lemmatisation
lemmas = []
for word, nltk_pos in nltk_pos_tags:
    wordnet_pos = nltk_pos_to_wordnet_pos(nltk_pos)
    if wordnet_pos is None:
        lemma = word # Si aucun tag POS correspondant, le mot est laissé tel quel
    else:
        lemma = lemmatizer.lemmatize(word, pos=wordnet_pos)
    lemmas.append(lemma)

# Filtrage des stopwords standard et des mots non informatifs supplémentaires
filtered_lemmas = [lemma for lemma in lemmas if lemma not in stop_words and lemma not in punctuation]

# Création des bigrams
bigrams_list = list(ngrams(lemmas, 2))
# Fusion des mots dans les bigrams avec un underscore pour une meilleure lisibilité
bigrams = ['_'.join(gram) for gram in bigrams_list]

# Création des trigrams
trigrams_list = list(ngrams(lemmas, 3))
# Fusion des mots dans les trigrams avec un underscore pour une meilleure lisibilité
trigrams = ['_'.join(gram) for gram in trigrams_list]

# Concaténation des tokens et des bigrams
return " ".join(filtered_lemmas + bigrams + trigrams)
```

2.1 Nettoyage des données

Synthèse du nettoyage des données

- Suppression des avis avec textes absents
- Nettoyage des textes
 - Suppression des stop words (mots vides)
 - Suppression des caractères spéciaux
 - Remplacement de certains mots (“service client” par “service_client” c'est-à-dire trouver des bigrammes et trigrammes)
 - Suppression des contractions
- Faire un export du fichier nettoyé pour ne plus à le refaire lors des prochaines sessions, ce fichier doit contenir les colonnes suivantes
 - Avis corrigés et traduits

2.2 Exploration Simple

Affichage des 100 mots les plus fréquents avec WordCloud



Analyse du WordCloud

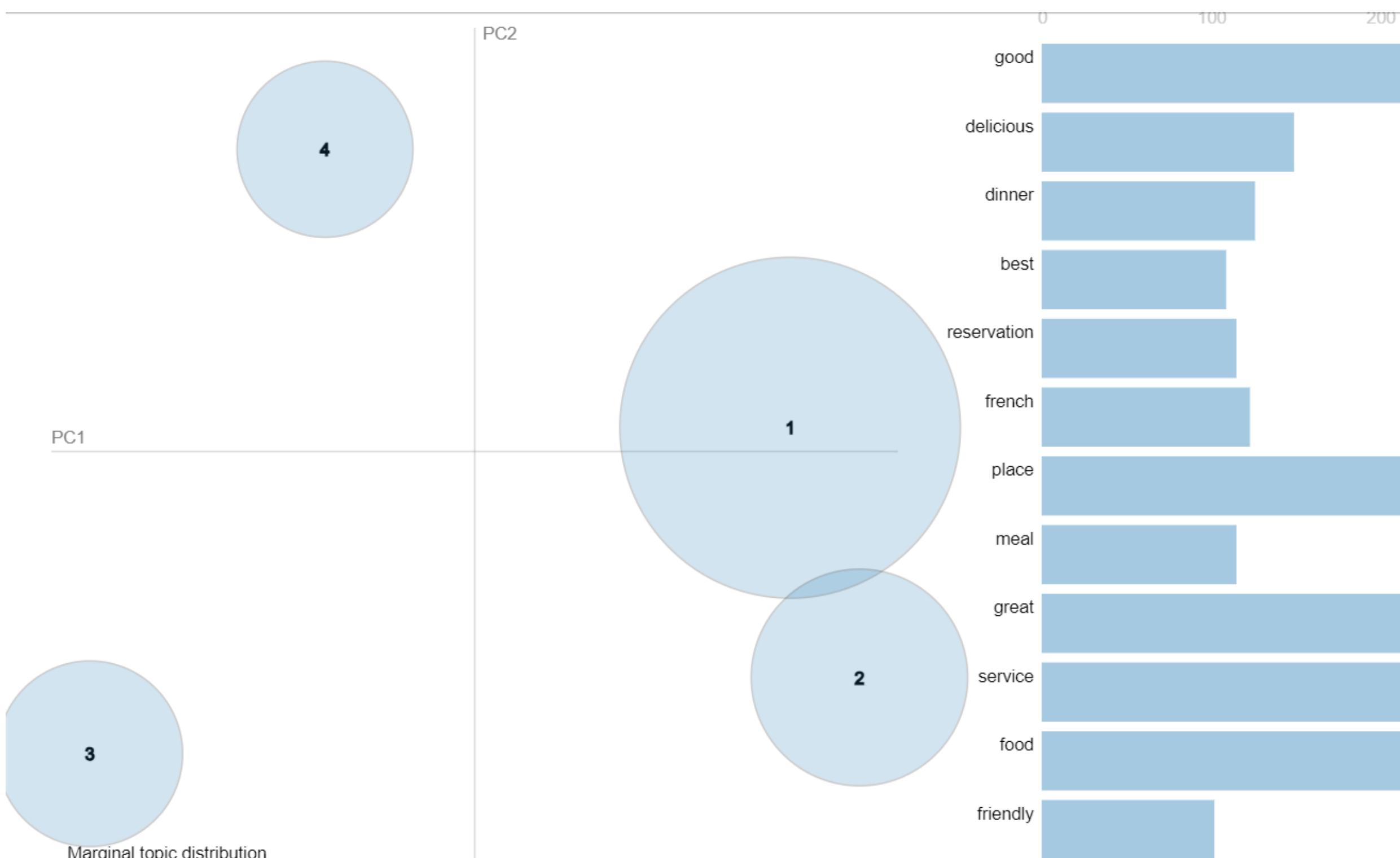
- Le preprocessing semble réussi car les mots ont l'air cohérents, on a enlevé tous les stopwords non pertinents
 - Identification des bi-grams et tri-grams cohérents

2.2 Exploration Simple

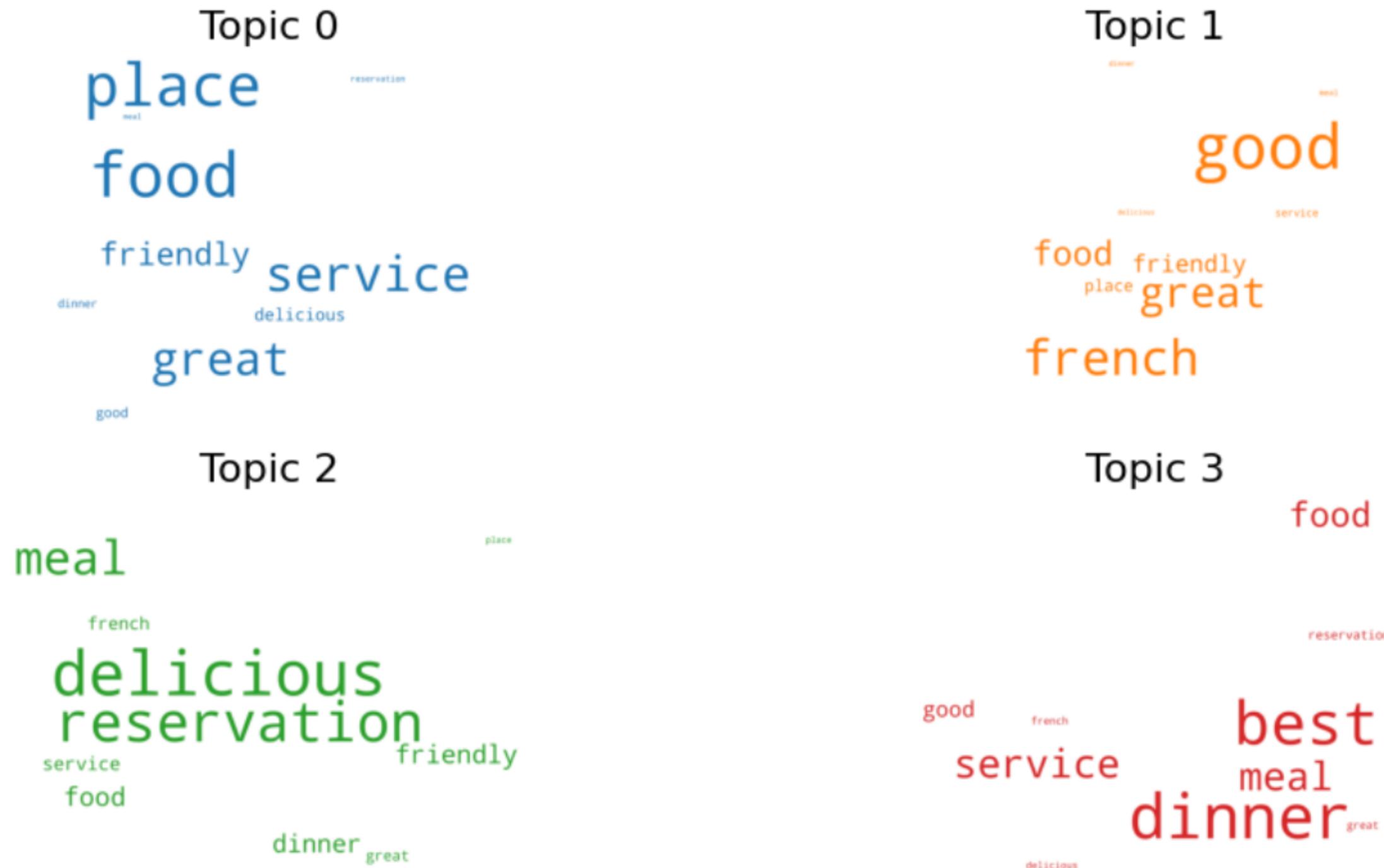
Topic Modeling LDA:

4 topics qui ressortent:

1. Recommended about Food & Service
2. Restaurants with Reservation
3. French Food
4. Restaurant for Dinner



2.2 Exploration Simple



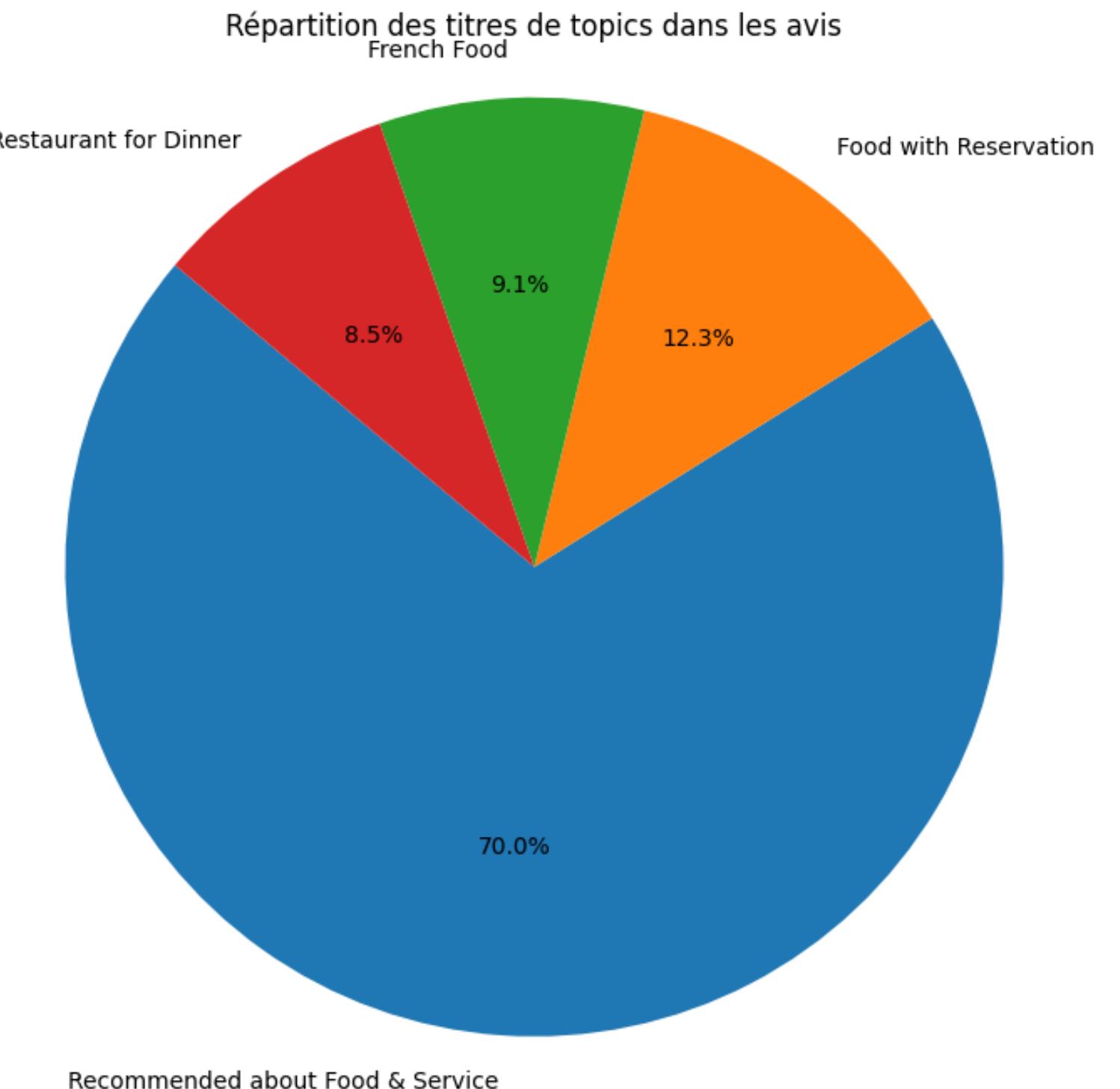
WordCloud de chaque Topics :

- 4 topics qui ressortent:
1. Recommended for Food & Service
 2. Restaurants with Reservation
 3. French Food
 4. Restaurant for Dinner

2.2 Exploration Simple

Visualisation de la proportion d'avis dans chaque topic

- La plupart des avis sont des recommandations de restaurants et services
- Les trois autres topics se valent en proportion.

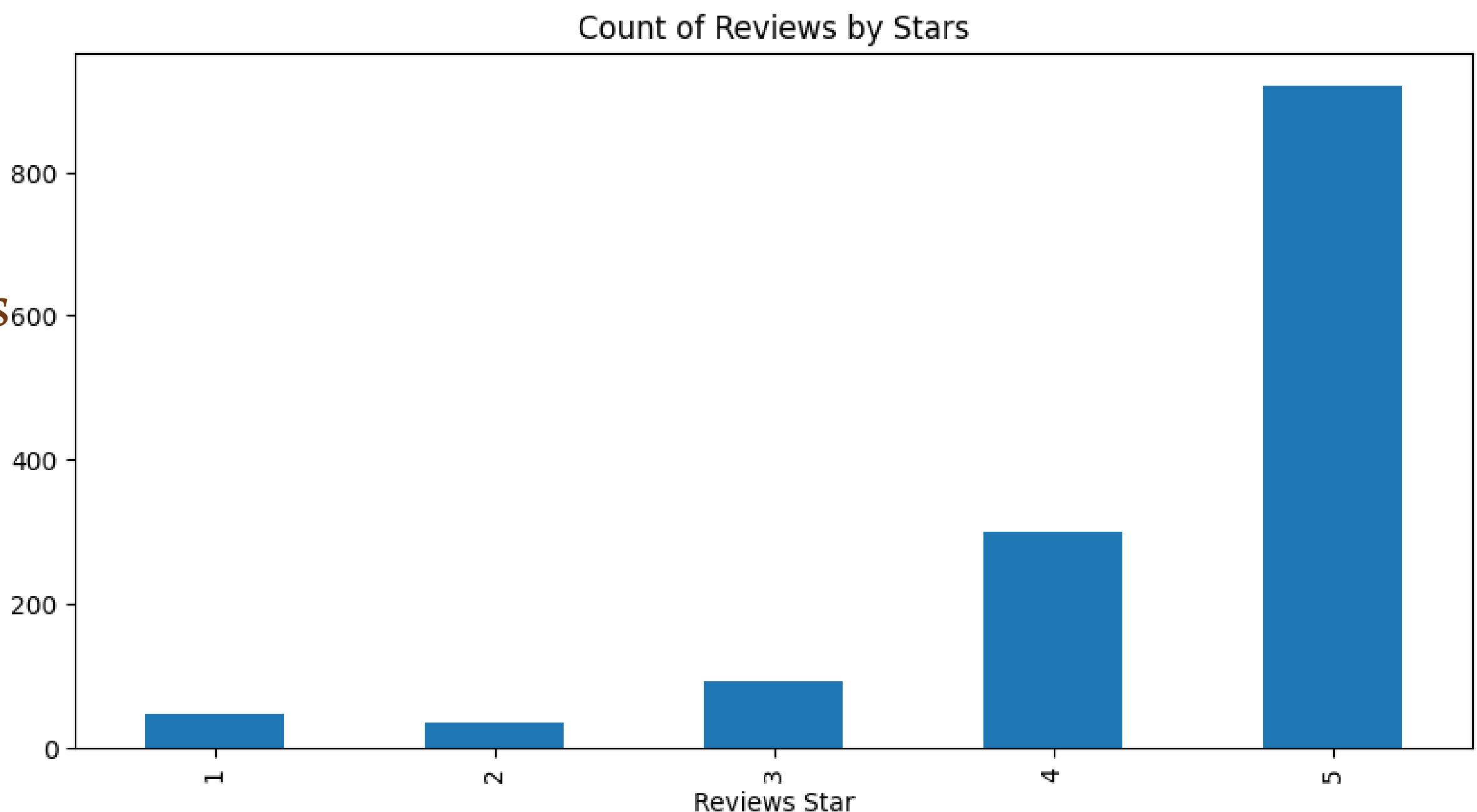


2.2 Exploration Simple

Visualisation de la répartition du nombre d'avis pour chaque note

- La plupart des avis sont des avis positifs noté 5.
- Les trois autres topics se valent

=> déséquilibre des classes



2.3

Embedding pour Identifier les mots similaires

Embedding avec Word2Vec

```
from gensim.models import Word2Vec
from sklearn.metrics.pairwise import cosine_similarity

# Préparation des données pour Word2Vec (tokenisation a déjà été faite)
texts = [text.split() for text in review_df['preprocessed_text']]

# Entraînement du modèle Word2Vec
w2v_model = Word2Vec(texts, vector_size=80, window=10, min_count=1, workers=4, epochs=30)

# Enregistrer le modèle pour une utilisation ultérieure
w2v_model.save("word2vec_model.model")
```

Après avoir préprocess les données, on va vectoriser le texte en utilisant le Word2Vec et nous l'enregistrons afin de ne plus avoir à l'exécuter.

2.3

Embedding pour Identifier les mots similaires

Embedding avec Word2Vec + Similarity research

```
# Trouver des mots similaires
similar_words = w2v_model.wv.most_similar('great_meal', topn=10)
print(similar_words)

✓ 0.0s
[('taste_menu', 0.9973956346511841), ('staff_friendly', 0.9973926544189453), ('price_reasonable', 0.9973801970481873

from scipy.spatial import distance

def euclidean_distance(word, model, top_n=10):
    word_vector = model.wv[word]
    all_word_vectors = np.array([model.wv[w] for w in model.wv.index_to_key if w != word])

    distances = [distance.euclidean(word_vector, wv) for wv in all_word_vectors]
    nearest_indices = np.argsort(distances)[:top_n]

    return [(model.wv.index_to_key[i], distances[i]) for i in nearest_indices]

# Exemple d'utilisation de la distance euclidienne
print(euclidean_distance('dinner', w2v_model))

✓ 0.2s
hard', 0.15678110718727112), ('house', 0.1579771488904953), ('might', 0.16201937198638916), ('fancy', 0.16890712082386017)]
```

On a utilisé 2 méthodes de similarity research :

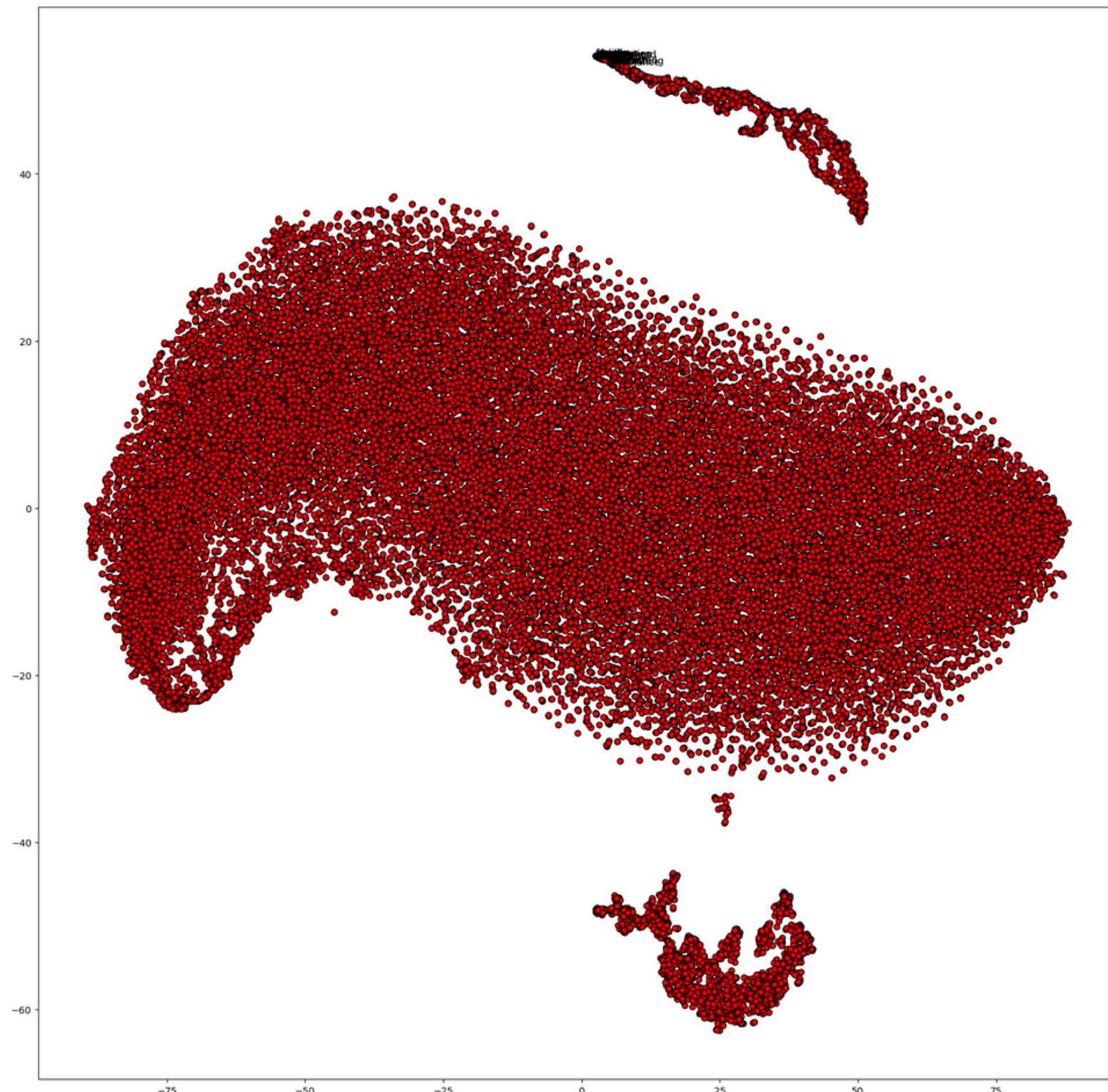
- inclus dans Word2Vec avec la fonction `most_similar`
- en utilisant la distance euclidienne, on associe les mots qui ont un sens proches car leur distances dans l'espace vectoriel est faible

2.3

Embedding pour Identifier les mots similaires

Visualisation avec Matplotlib

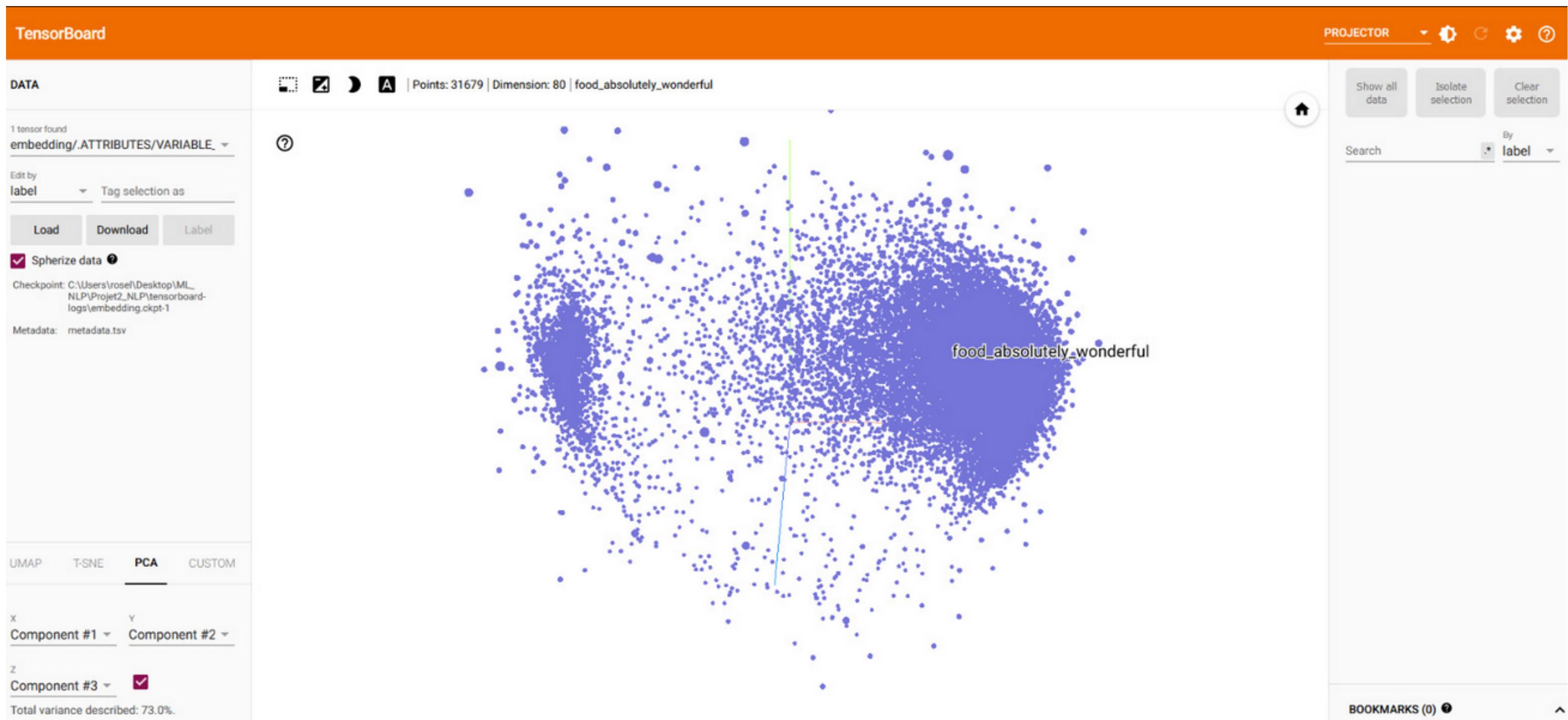
- Chaque point sur le graphique représente un mot (ou un vecteur de mots) de notre modèle Word2Vec.
- Les mots sont regroupés par similarité sémantique.



2.3

Embedding pour Identifier les mots similaires

Visualisation avec TensorBoard

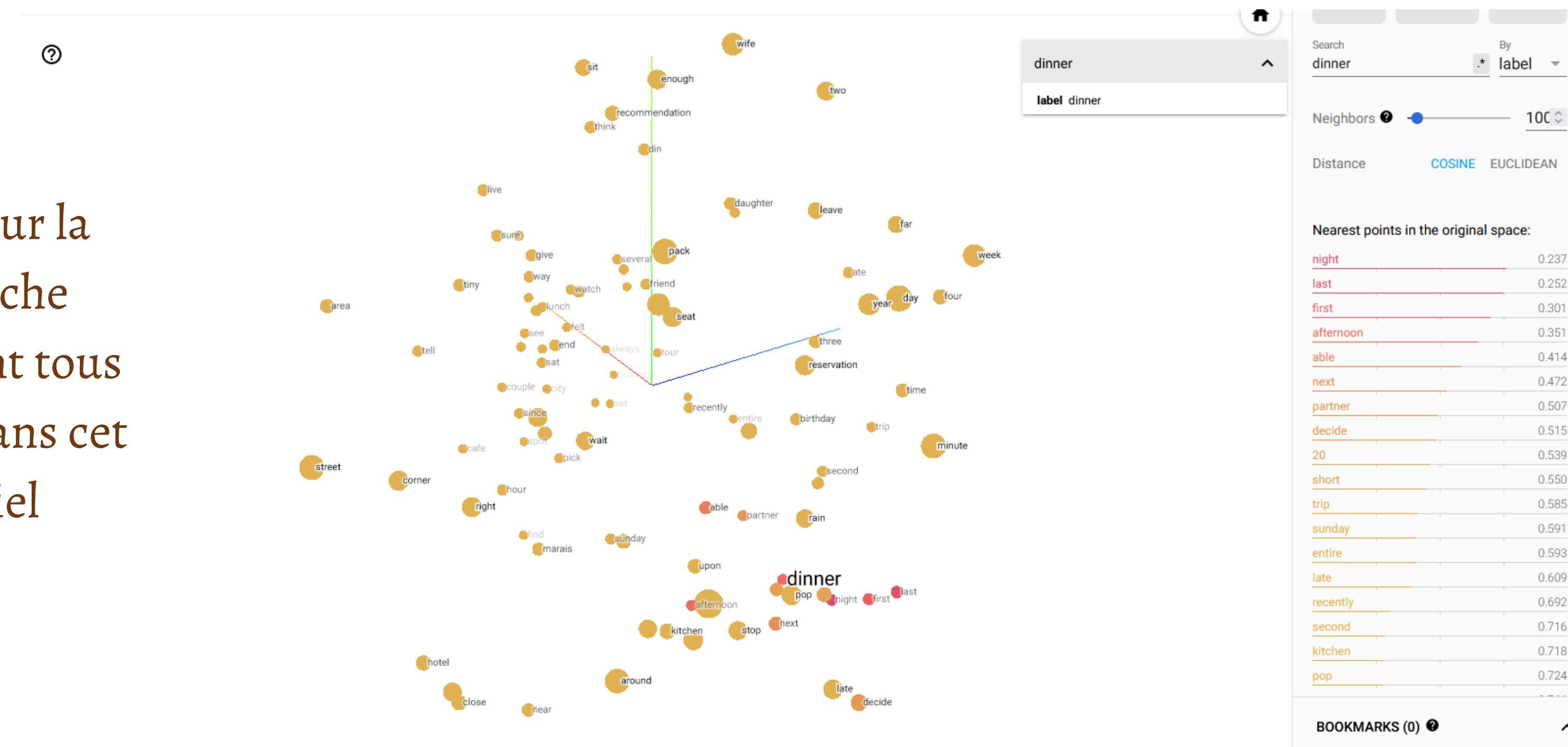


2.3

Embedding pour Identifier les mots similaires

Visualisation avec TensorBoard

Lorsqu'on tape sur la barre de recherche ‘dinner’, on obtient tous les mots voisins dans cet espace vectoriel



2.3

Embedding pour Identifier les mots similaires

Semantic research

```
def semantic_search(query, model, texts, top_n=5):
    # Tokeniser et calculer la moyenne des vecteurs de mots pour la requête
    query_tokens = query.split()
    query_vector = average_word_vectors(query_tokens, model, set(model.wv.index_to_key), model.vector_size)

    # Calculer la moyenne des vecteurs de mots pour chaque document
    docs_vectors = [average_word_vectors(text.split(), model, set(model.wv.index_to_key), model.vector_size) for text in texts]

    # Calculer la similarité cosinus entre la requête et chaque document
    similarities = cosine_similarity([query_vector], docs_vectors).flatten()
    similar_indices = similarities.argsort()[-top_n:][::-1]
    similar_texts = [(texts[i], similarities[i]) for i in similar_indices]

    return similar_texts

# Exemple de recherche sémantique
query = "great food and service"
print(semantic_search(query, w2v_model, review_df['preprocessed_text'].tolist()))
3.4s
wonderful food attentive friendly knowledgeable service fromage plate delicious wonderful_food food_attentive_attentive_fr
```

On obtient un score de
0.9998014594468111



Python

2.4

Features ajoutés

preprocessed_text	Topic_Title	label
base menu present write glow review classiques...	Food with Reservation	2
place feel little bit tourist trap people din ...	Recommended about Food & Service	2
love french onion soup try comptoir la gastron...	Food with Reservation	2
daughter son inlaw mom gratin delicious table ...	French Food	2

```
# Convertir les évaluations en labels de sentiment
def convert_rating_to_sentiment(rating):
    if rating < 3:
        return 0 # négatif
    elif rating > 3:
        return 2 # positif
    else:
        return 1 # neutre

# Appliquer la conversion sur la colonne de rating
review_df['label'] = review_df['rating'].apply(convert_rating_to_sentiment)
```

Nous avons ajouté les topics sur chaque avis et également le label. On considère que la note (rating) 3 correspond au sentiment neutre (1), en dessous correspond à un avis négatif (0) et au dessus de 3 à un avis positif(2).

3 Modélisation et interprétation

1. Problème rencontré
2. Nettoyage du nouveau dataset pour l'entraînement
3. VADER
4. Machine Learning classique
5. CNN
6. Transformers: distilBert analyse de sentiment
7. Transformers: ABSA avec setfit
8. LSA (Latent Semantic Analysis) pour les résumés automatiques
9. LLM pour la génération de texte

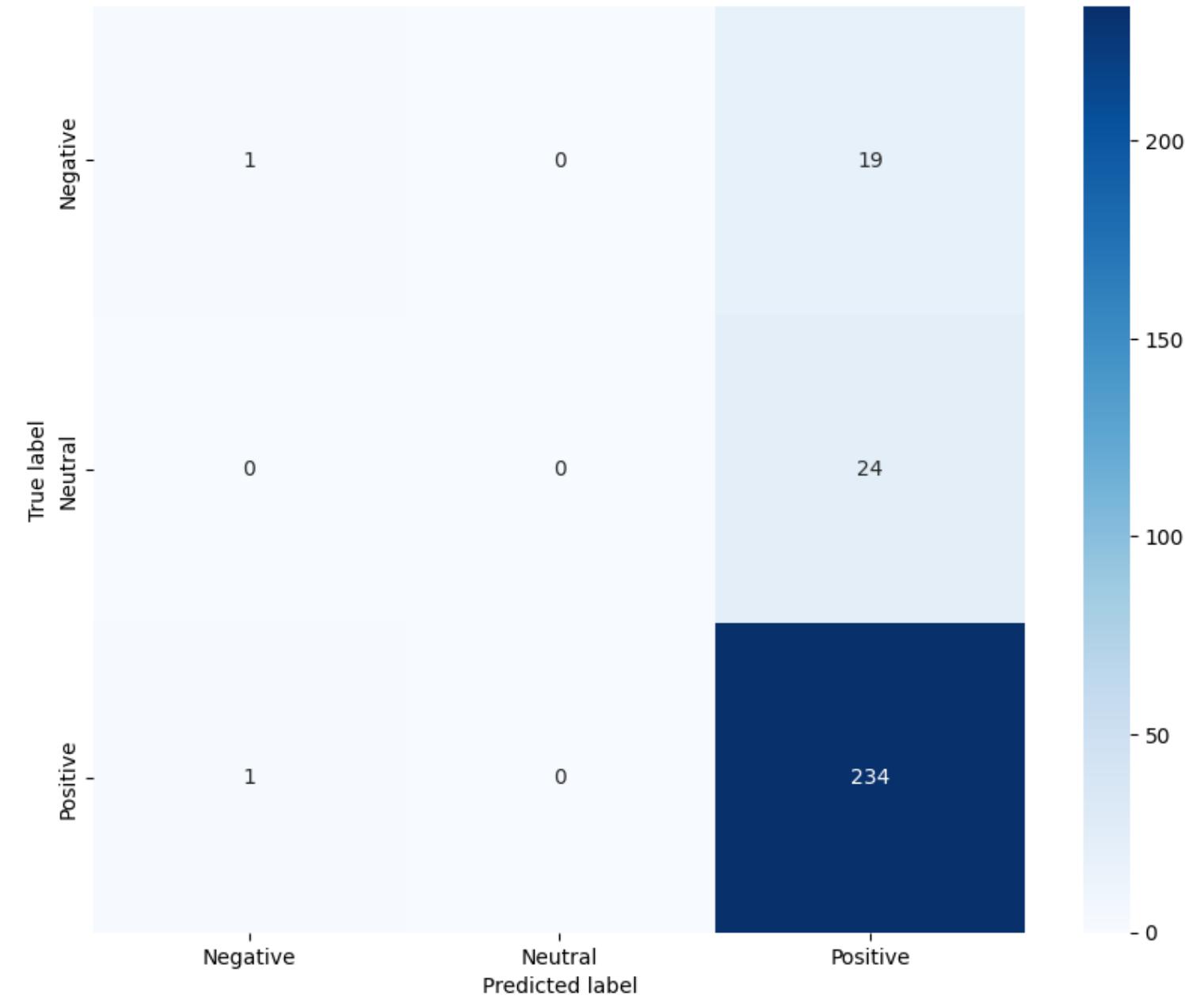
3.1 Problème rencontré

Nous nous sommes rendus compte que notre dataset de 1393 avis était très déséquilibré. En effet, on avait 81 avis négatifs pour 1220 avis positifs. Nous avons fait quelques modèles mais peu concluant au vu de la matrice de confusion:

- on prédisait plus de faux positifs que de vrais négatifs

Ainsi nous avons voulu entraîner notre modèle sur un dataset plus large trouvé sur kaggle:

- <https://www.kaggle.com/datasets/farukalam/yelp-restaurant-reviews/data>



3.2 Nettoyage du nouveau dataset pour l'entraînement

```
# Chemin du fichier JSON
data_file_path = "yelp_academic_dataset_review.json"

# Initialiser une liste pour stocker les données
data = []

# Ouvrir le fichier et lire les lignes
with open(data_file_path, 'r', encoding='utf-8') as data_file:
    for i, line in enumerate(data_file):
        # Convertir la ligne de JSON en dictionnaire et l'ajouter à la liste 'data'
        data.append(json.loads(line))

        # Arrêter après avoir lu les 10 000 premiers éléments
        if i + 1 == 100000:
            break

# Créer un DataFrame à partir des données
data_train_review = pd.DataFrame(data)

# Afficher les premières lignes du DataFrame pour vérifier
data_train_review.tail()
```

```
# Appliquer la fonction de prétraitement sur chaque avis
data_train_review['preprocessed_text'] = data_train_review['text'].apply(preprocess_text)
```

Ce dataset se compose d'avis sur des restaurants spécifiques de Yelp. Les données couvrent une période de plus de 15 ans. On applique la fonction de preprocessing et on prend les 100 000 avis de ce dataset

3.2 Nettoyage du nouveau dataset pour l'entraînement

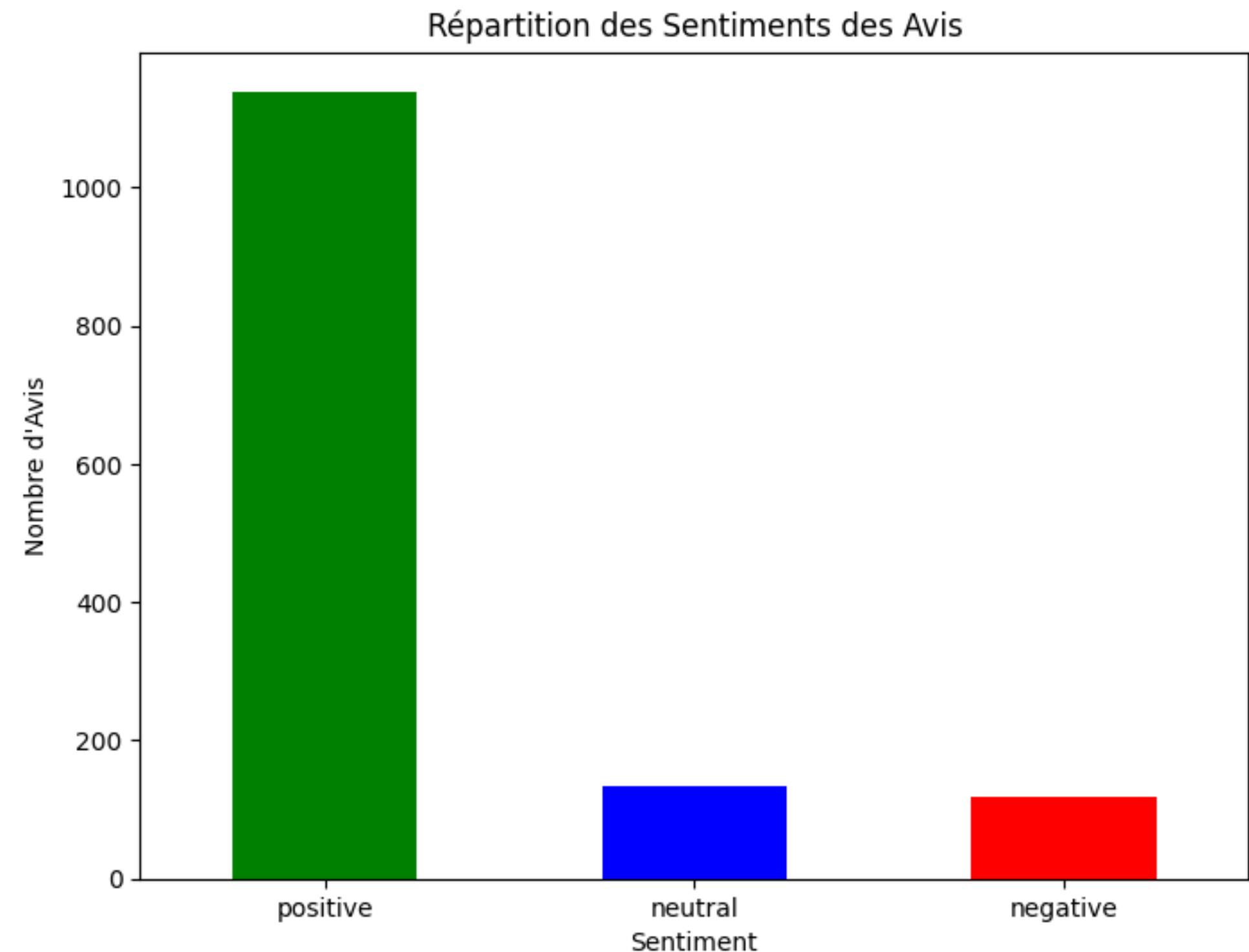
On affiche les 100 mots les plus fréquents et nous faisons du topic modeling:

- Food Service
 - Recommended about Fast Food
 - Recommended about Food & service
 - Restaurant for Dinner



3.3 Vader

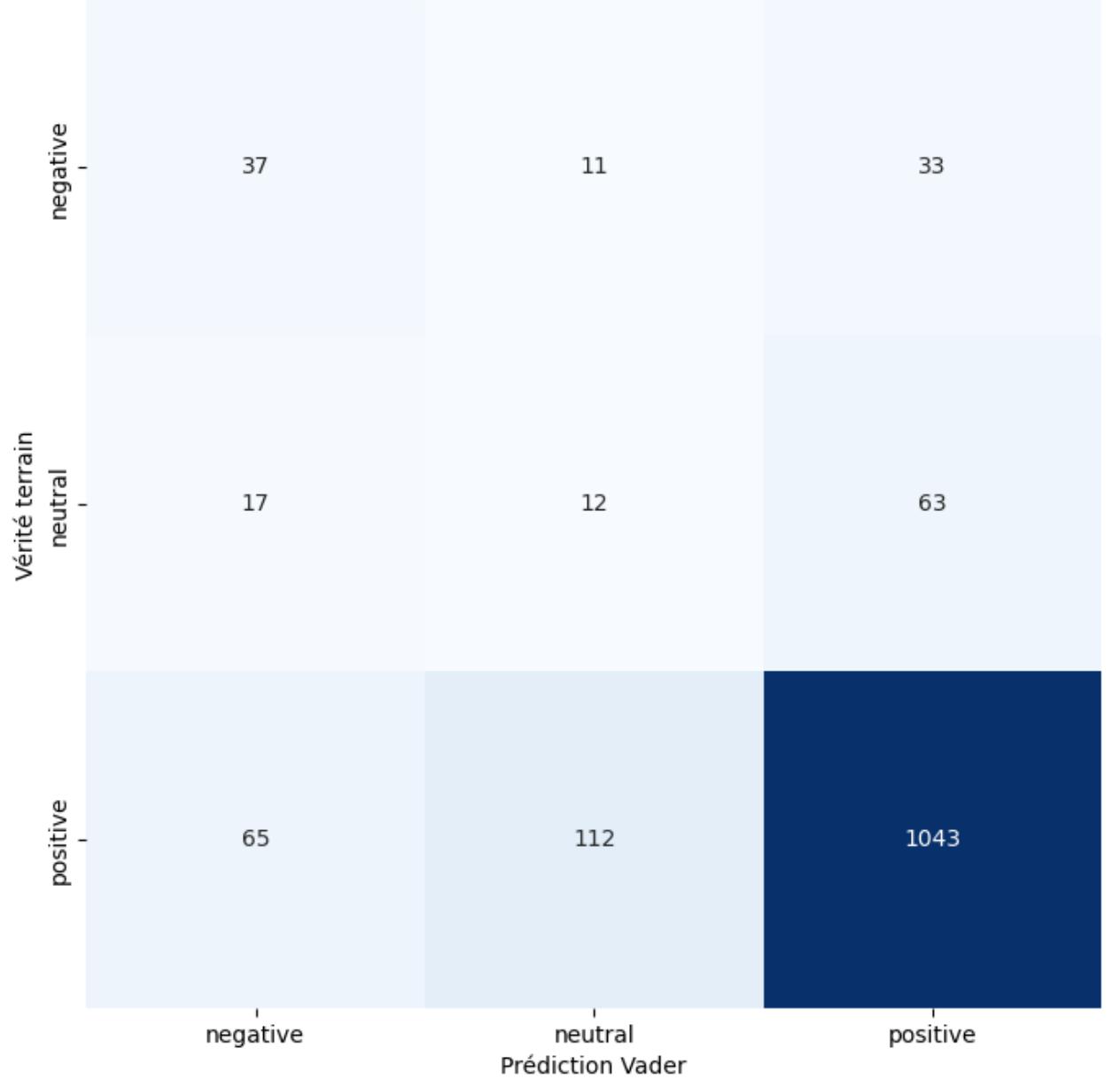
Pour commencer, on a simplement prédit le sentiment globale de chaque avis en utilisant Vader, qui est un analyseur lexical utilisé pour attribuer des scores de sentiment à une phrase. Il donne un score de polarité et nous avons défini que si la prédiction était supérieure ou égale à 0.05 alors nous le considérons comme positive, en-dessous de -0.05 négative et sinon neutre.



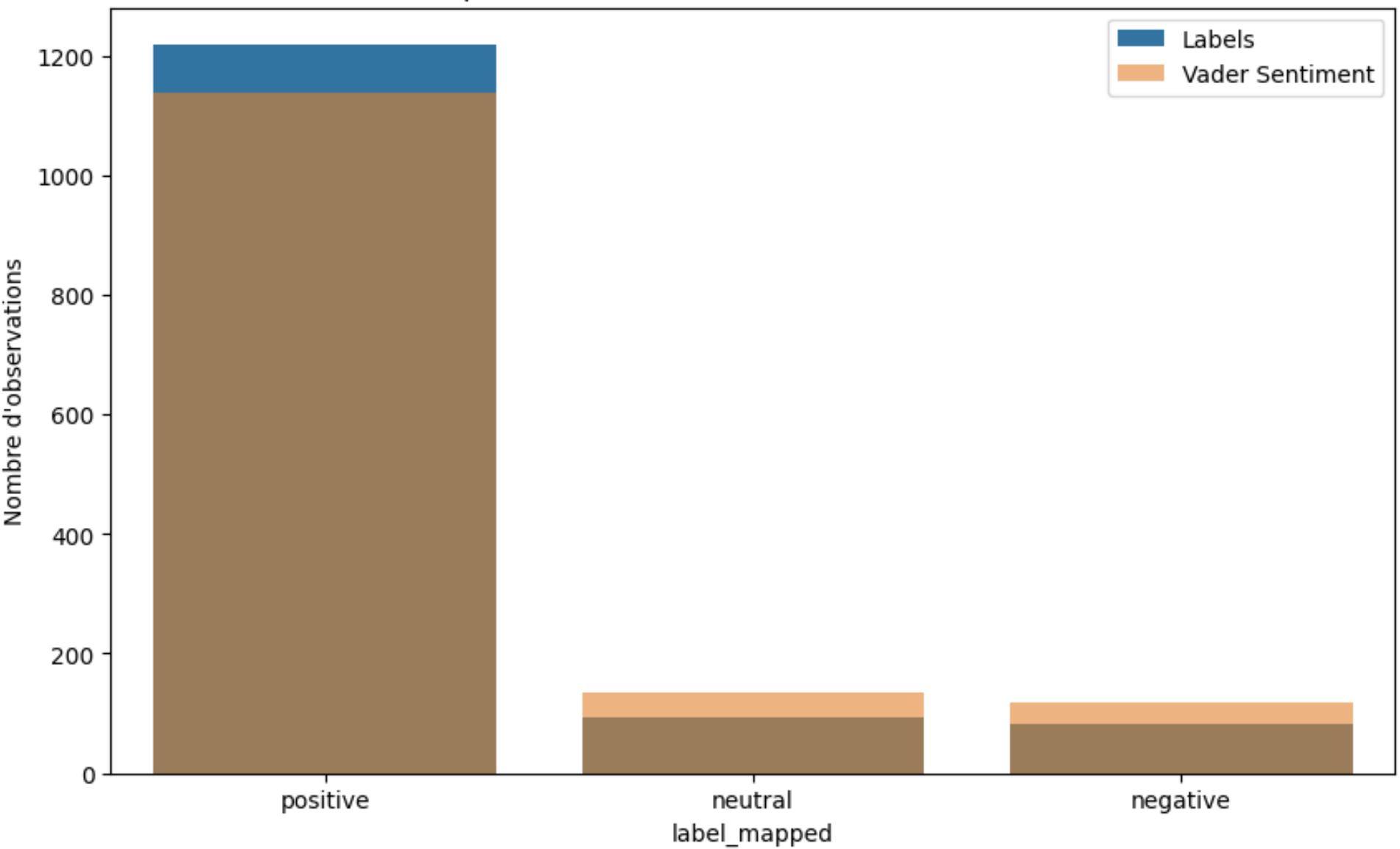
Vader prédit beaucoup plus de positif que les autres sentiments

3.3 Vader

Matrice de confusion pour la prédiction de Vader



Comparaison entre les labels et le sentiment Vader



3.3 Vader

Analyse du matrice de confusion:

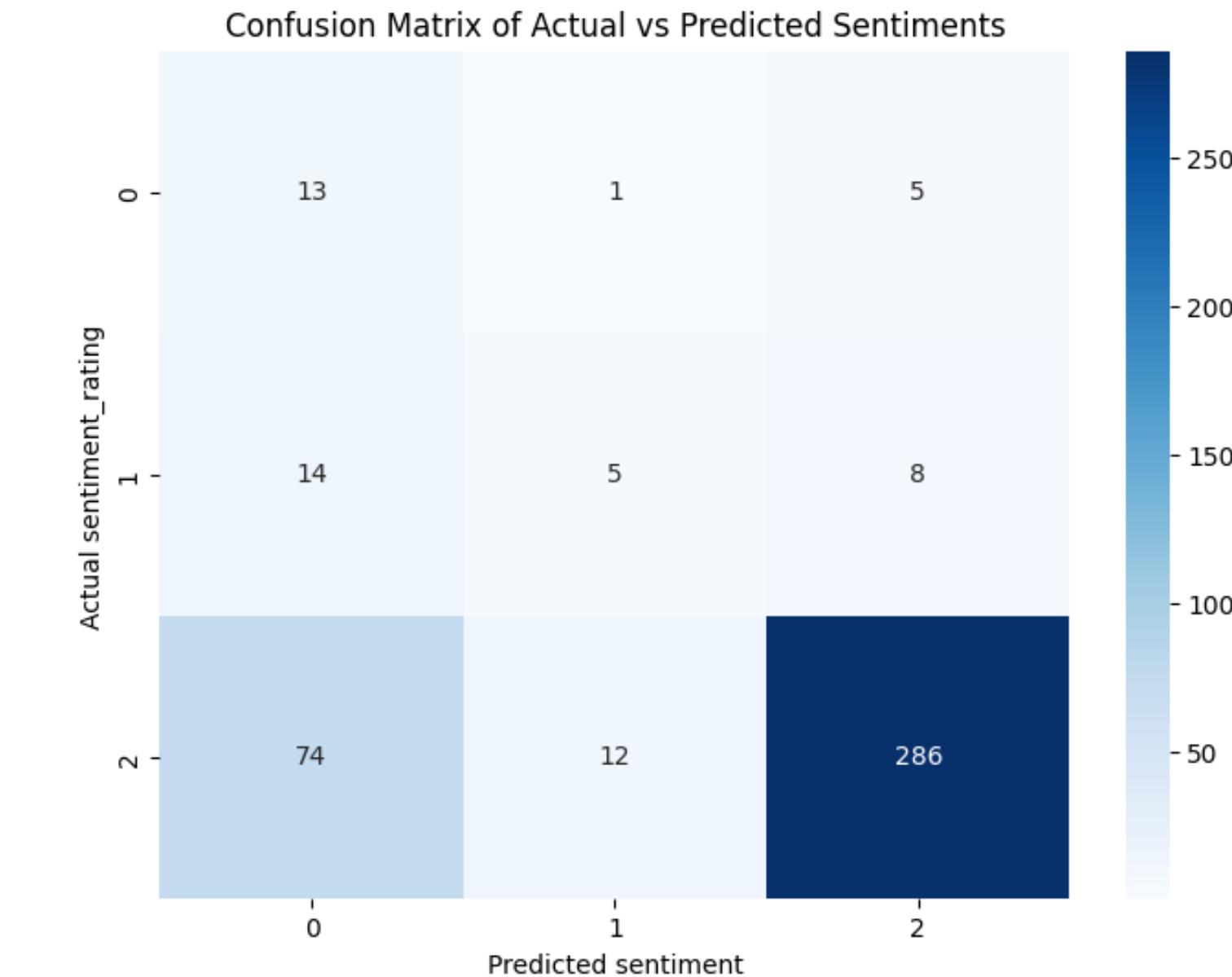
- VADER tend à classer les avis comme positifs, comme en témoigne le nombre élevé de faux positifs (33 pour les vrais négatifs et 63 pour les vrais neutres).
 - Le nombre de faux négatifs est relativement faible, ce qui indique que lorsque VADER prédit un avis comme négatif, il est plus susceptible d'être correct.
- => Dans l'ensemble, cette matrice indique que VADER est plus enclin à classer un avis comme positif, ce qui peut être dû à un biais dans les données car nos données sont très déséquilibrées.

3.4

Machine Learning: Easy Ensemble Logistic regression + SVC

1. Vectorisation TF-IDF : Les textes sont transformés en une matrice TF-IDF, qui reflète l'importance des mots dans les documents en fonction de leur fréquence dans le document et leur rareté dans l'ensemble du corpus.

2. Modèle de Machine Learning : EasyEnsemble Classifier, pour créer un ensemble de classificateurs en échantillonnant de manière équilibrée pour pallier au déséquilibre des classes. Les modèles considérés sont un modèle de régression logistique et une machine à vecteurs de support (SVC, Support Vector Classifier).



	precision	recall	f1-score	support
negatif	0.13	0.68	0.22	19
neutre	0.28	0.19	0.22	27
positif	0.96	0.77	0.85	372
accuracy			0.73	418
macro avg	0.45	0.55	0.43	418
weighted avg	0.88	0.73	0.78	418

3.4

Machine Learning: Easy Ensemble Logistic regression + SVC

1. Vectorisation TF-IDF

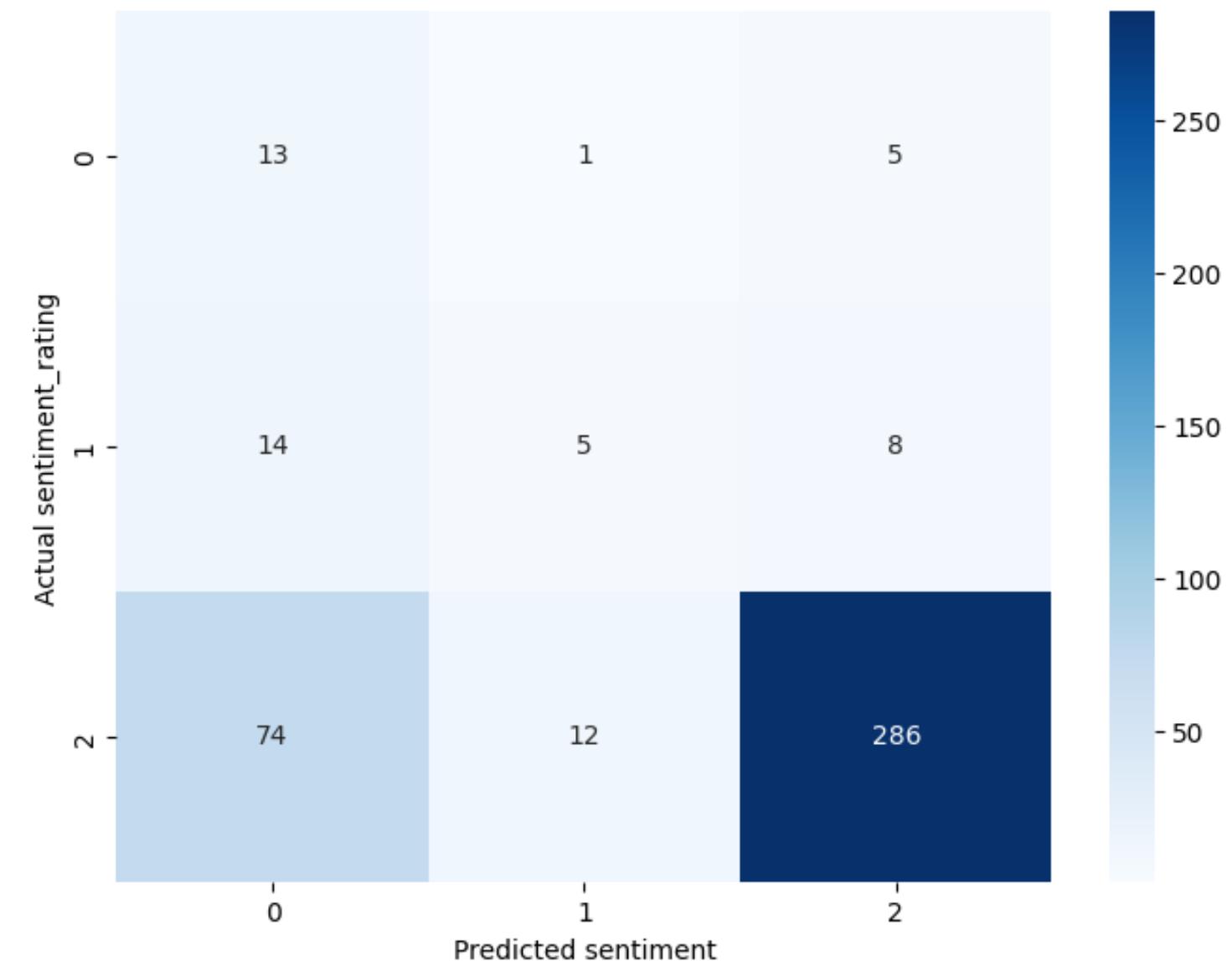
2. Modèle de Machine Learning : EasyEnsemble Classifier

Résultats: Le rapport de classification montre une précision globale de 88%. Cependant, le modèle est biaisé vers la classe 'positif' (372 cas) et performe mal sur les classes 'négatif' (19 cas) et 'neutre' (27 cas), comme le reflète la matrice de confusion et les faibles scores F1 pour ces classes.

Bien que l'exactitude globale soit élevée, elle est principalement influencée par la capacité du modèle à reconnaître les avis positifs, qui sont majoritaires.

	precision	recall	f1-score	support
negatif	0.13	0.68	0.22	19
neutre	0.28	0.19	0.22	27
positif	0.96	0.77	0.85	372
accuracy			0.73	418
macro avg	0.45	0.55	0.43	418
weighted avg	0.88	0.73	0.78	418

Confusion Matrix of Actual vs Predicted Sentiments



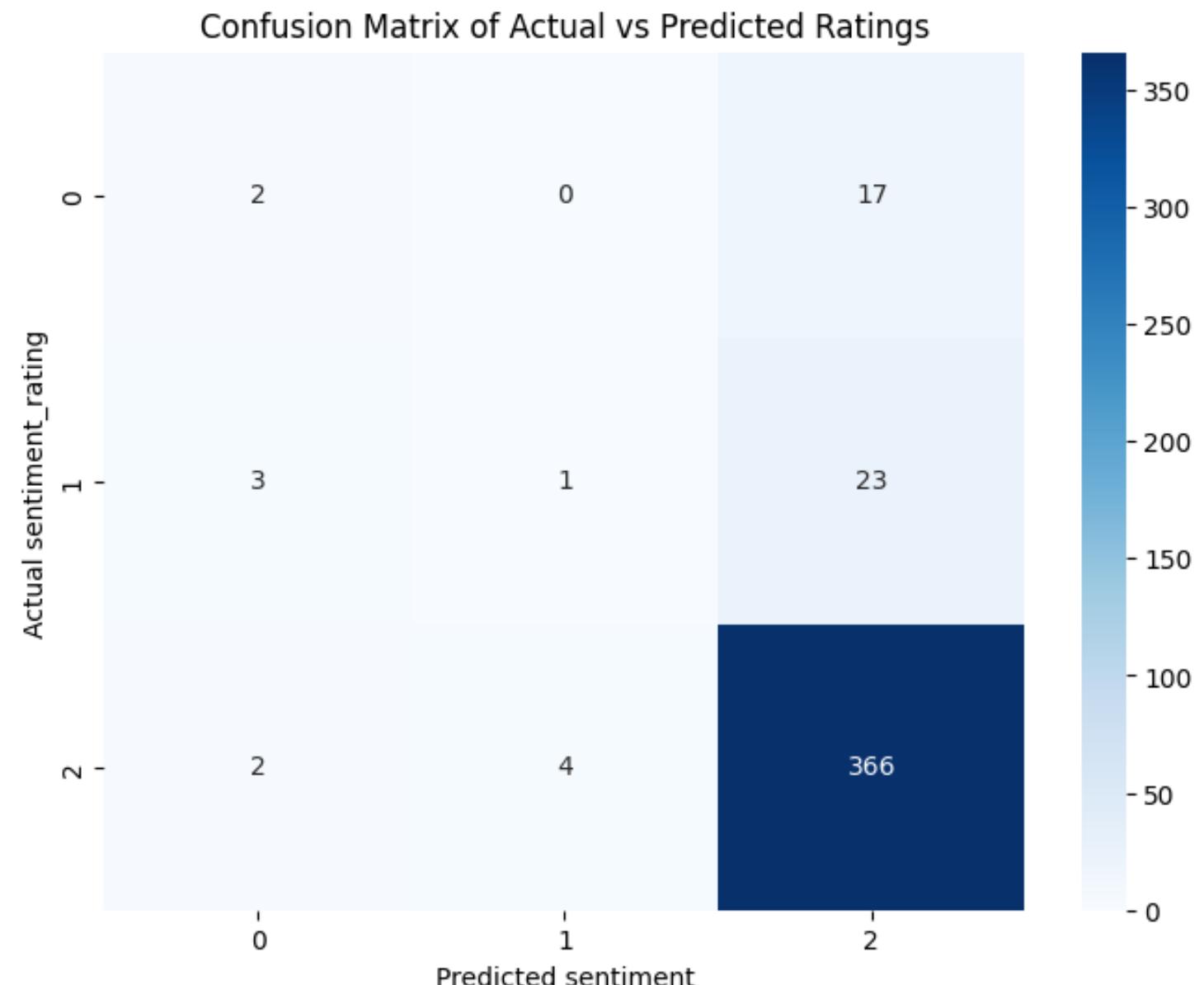
3.4 Machine Learning

1. Vectorisation TF-IDF : On réutilise la même transformation de données.

2. Modèle de Machine Learning : Un modèle de forêt aléatoire (RandomForestClassifier) est entraîné sur les caractéristiques TF-IDF de l'ensemble d'entraînement pour prédire les scores de sentiment.

Résultats: On observe facilement que le modèle n'est pas assez performant pour les classes non positives.

	precision	recall	f1-score	support
negatif	0.29	0.11	0.15	19
neutre	0.20	0.04	0.06	27
positif	0.90	0.98	0.94	372
accuracy			0.88	418
macro avg	0.46	0.38	0.39	418
weighted avg	0.83	0.88	0.85	418



3.5 CNN

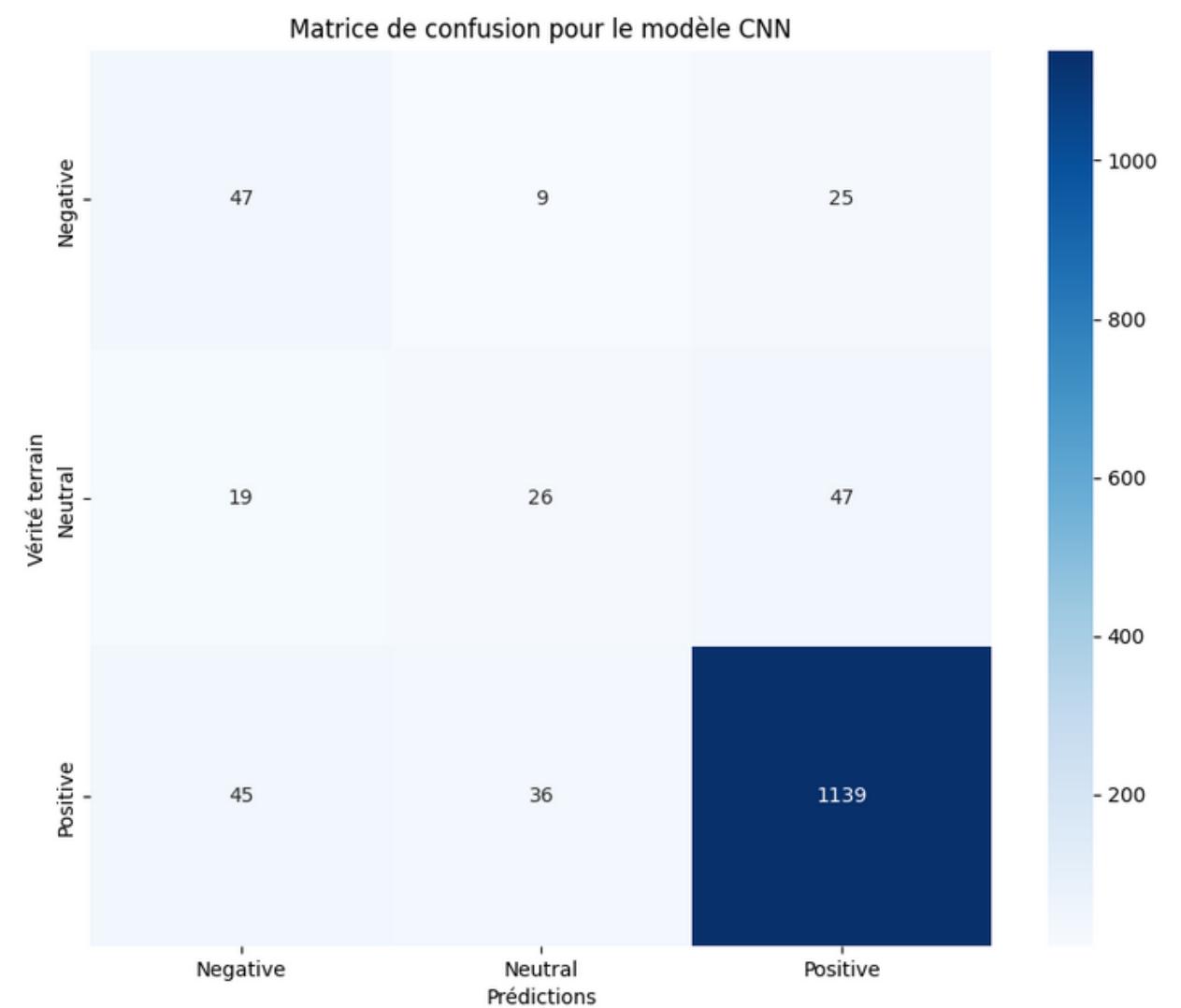
Performance sur les Avis Positifs :

- Le modèle excelle dans la classification des avis positifs, avec une grande majorité d'avis correctement identifiée (1139 sur 1220).
- Cela se reflète dans la précision élevée (0.94), le rappel élevé (0.93), et le score F1 élevé (0.94) pour cette classe.

Performance sur les Avis Négatifs et Neutres:

- La performance est modérée pour les avis négatifs avec 47 prédictions correctes sur 81 et sur les avis neutres avec seulement 26 prédictions correctes sur 92.

	precision	recall	f1-score	support
Negative	0.42	0.58	0.49	81
Neutral	0.37	0.28	0.32	92
Positive	0.94	0.93	0.94	1220
accuracy			0.87	1393
macro avg	0.58	0.60	0.58	1393
weighted avg	0.87	0.87	0.87	1393



3.5 CNN

Analyse du CNN

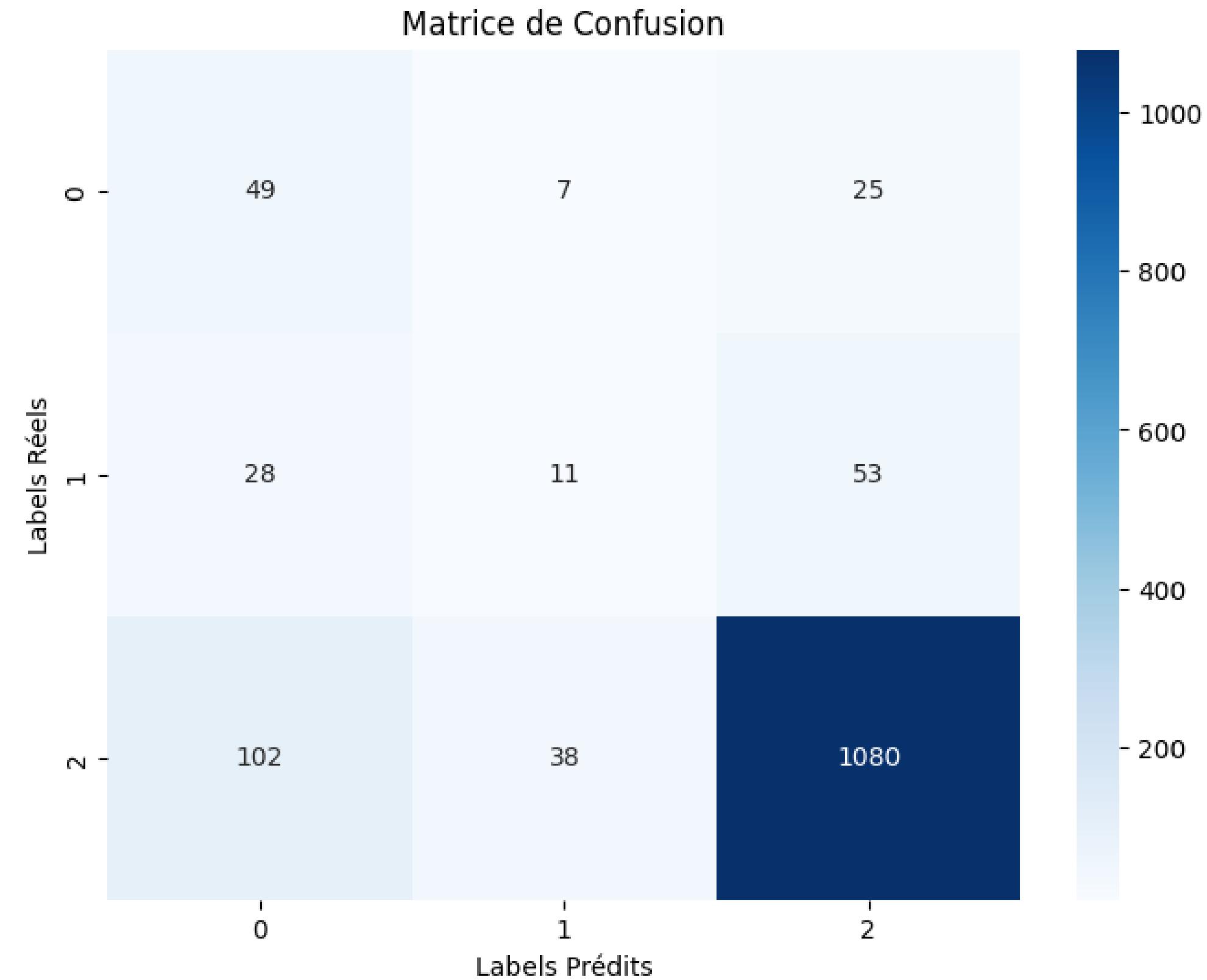
- Le modèle est biaisé en faveur des avis positifs en effet les avis positifs représentent la plupart des avis .
- Les avis négatifs et neutres sont moins bien prédits, ce qui peut être dû à un déséquilibre des classes, et à des caractéristiques moins distinctes des textes.
- L'accuracy globale du modèle est de 0.87, ce qui peut sembler élevé, mais cette métrique est trompeuse car on est en présence d'un déséquilibre de classe.

3.6

Transformers distilBERT

On utilise une pipeline de classification de texte de Hugging Face Transformers pour analyser le sentiment des avis. Le modèle utilisé est spécialement préentraîné sur des avis de restaurants Yelp, ce qui devrait fournir une analyse précise pour des données similaires

Résultat: on observe plus de prédiction dans les classes neutres et négatives



["mrcaelumn/yelp_restaurant_review_sentiment_analysis"](#)

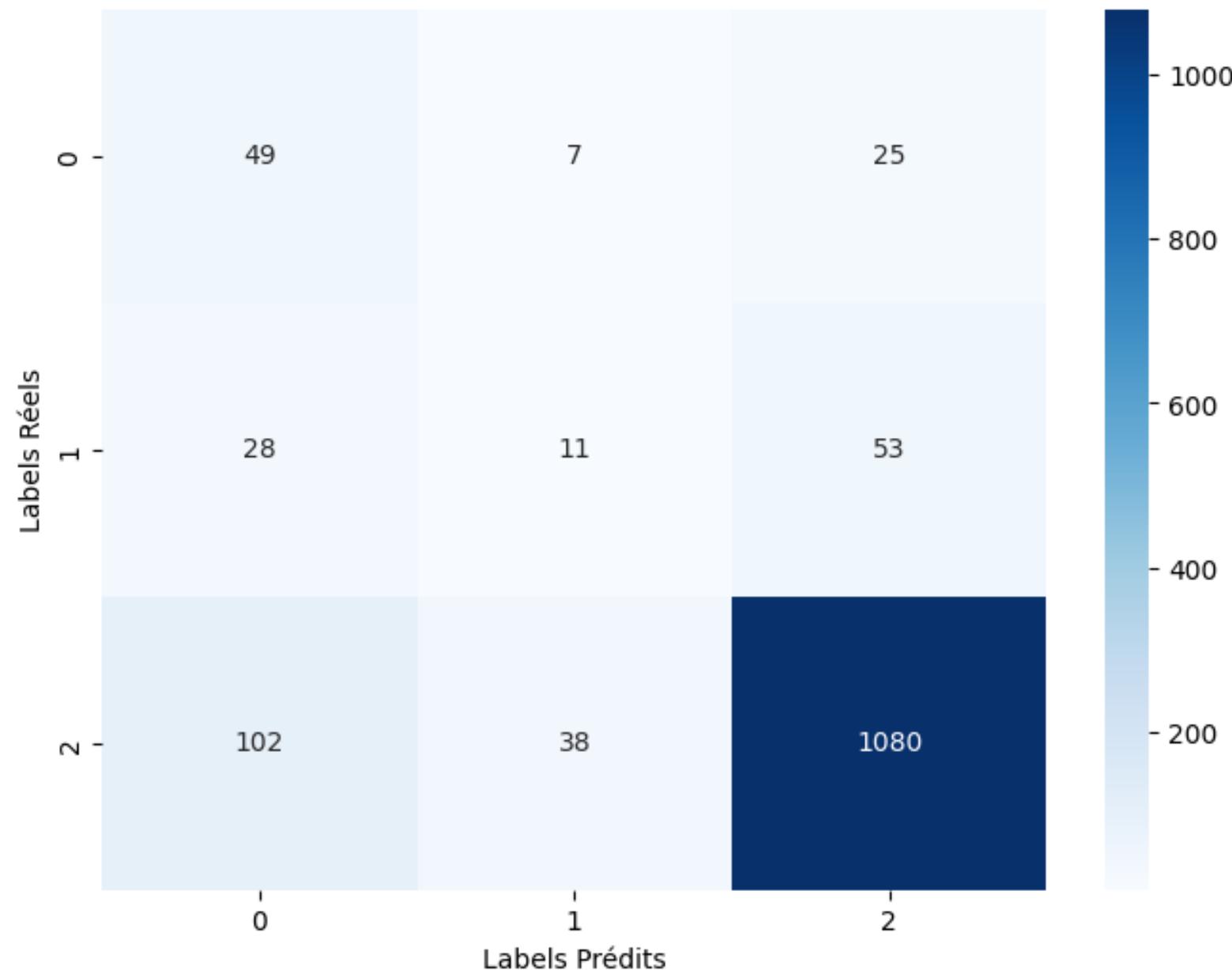
3.6

Transformers distilBERT

Résultats: Le modèle est très performant pour la classe 'Positive' avec une précision de 93% et un score F1 de 90%, mais il est moins précis pour les classes 'Negative' et 'Neutral'. Malgré une précision plus basse pour ces dernières, le rappel pour 'Negative' est relativement élevé (62%), ce qui indique que le modèle est capable d'identifier une grande partie des avis négatifs, bien qu'avec un nombre significatif de faux positifs. On obtient une amélioration dans la prédictions des classes minoritaires bien que les performances sur celles ci soient toujours insuffisantes.

		precision	recall	f1-score	support
	Negative	0.25	0.62	0.36	81
	Neutral	0.20	0.12	0.15	92
	Positive	0.93	0.87	0.90	1220
	accuracy			0.81	1393
	macro avg	0.46	0.54	0.47	1393
	weighted avg	0.84	0.81	0.82	1393

Matrice de Confusion



3.7

Transformers ABSA

L'ABSA (Aspect-Based Sentiment Analysis) est une sous-discipline de l'analyse des sentiments qui se concentre non seulement sur la détection de la polarité générale (positive, négative, neutre) d'un texte donné, mais aussi sur l'identification des différents aspects ou caractéristiques spécifiques mentionnés dans le texte et l'analyse des sentiments associées à chaque aspect spécifique.

On utilise un modèle pré-entraîné sur des avis de restaurants, cela nous permet de décomposer les avis pour pouvoir prendre en compte plusieurs sentiments évoqués (ambiance/food/service).

"The ambiance was lovely as well as the service , but the food was not that good."

```
[["span": "ambiance", "polarity": "positive"},  
 {"span": "service", "polarity": "positive"},  
 {"span": "food", "polarity": "negative"}]]
```

"tomaarsen/setfit-absa-paraphrase-mpnet-base-v2-restaurants-aspect"

3.8

LSA (Latent Semantic Analysis)

Le but ici est de **produire un résumé concis** de l'ensemble des avis correspondant à un restaurant, ce qui peut aider à saisir rapidement l'opinion des clients notamment sur une période définie en cas de variation de la note récemment, on aurait une idée globale des avis sans avoir à les parcourir entièrement.

```
from sumy.parsers.plaintext import PlaintextParser
from sumy.nlp.tokenizers import Tokenizer
from sumy.summarizers.lsa import LsaSummarizer as Summarizer
def summarize_text(text, sentences_count=2):
    parser = PlaintextParser.from_string(text, Tokenizer("english"))
    summarizer = Summarizer()
    summary = summarizer(parser.document, sentences_count)
    return ' '.join([str(sentence) for sentence in summary])
# Regrouper les avis par 'business_id'
grouped_reviews = review_df.groupby('business_id')['translated_text'].apply(' '.join)
# Résumer les avis pour chaque restaurant
summarized_reviews = grouped_reviews.apply(lambda x: summarize_text(x, sentences_count=2))
# Créer un nouveau DataFrame pour les résumés
summarized_df = pd.DataFrame({'business_id': summarized_reviews.index, 'reviews': summarized_reviews})
```

On utilise une technique d'**analyse sémantique latente** pour résumer des textes. Il extrait les informations essentielles d'un texte en identifiant les relations sémantiques entre les mots et les phrases. Cette technique permet de créer des résumés qui capturent le sens global du texte.

3.8 LSA (Latent Semantic Analysis)

EXAMPLE:

Original Text: Got in early during opening time for lunch with a reservation. We were seated at one of the booths that sat four in the rear dining section of this limited... This place was highly recommended by youtube food bloggers if you're looking for an authentic french cuisine that the locals patronize. Reserved way ahead... We had a great time at Chez Savy. They had roomFor our large party last minute and we're so friendly and welcoming the waiter was especially great and...

Summary: This place was highly recommended by youtube food bloggers if you're looking for an authentic french cuisine that the locals patronize. Reserved way ahead... We had a great time at Chez Savy.

3.9

LLM GPT2 pour la génération de texte

1. Initialisation du modèle GPT-2 : On crée un pipeline de génération de texte en utilisant le modèle pré-entraîné 'gpt2' et on crée une fonction generate_insights.

2. Fonction generate_insights :

- Cette fonction prend en entrée une liste d'avis.
- Les avis sont combinés en un seul texte (combined_reviews).
- Un prompt est créé pour demander au modèle GPT-2 de générer des insights basés sur les avis. Il est formulé pour suggérer au modèle de détecter les atouts d'un restaurant à partir des commentaires.

```
# Initialiser le modèle GPT-2
generator = pipeline('text-generation', model='gpt2')
def generate_insights(reviews):
    combined_reviews = " ".join(reviews)
    prompt = (f"Here are some customer reviews on a restaurant deduct the restaurant's asset's.\n\n{combined_reviews}\n\n")

    summary = generator(prompt,max_length=250, num_return_sequences=1)
    return summary[0]['generated_text']
# Grouper les avis par 'business_id'
grouped_reviews = df.groupby('business_id')['text'].apply(list)
# Générer des insights pour chaque groupe d'avis
insights_by_business = grouped_reviews.apply(generate_insights)
# Afficher les insights
for business_id, insights in insights_by_business.items():
    print(f"Business ID: {business_id}\nInsights:\n{insights}\n")
```

3.9

LLM GPT2 pour la génération de texte

Here are some customer reviews on a restaurant deduct the restaurant's asset's.

Prompt soumis:

Great service but the food was bland. Loved the ambiance and the dessert, but the main course was too salty.

Réponse générée:

Great service but the food was bland. Loved the ambiance and the dessert, but the main course was too salty.

Great menu, great food! The service was good, the seating was good, the food was good, the drink is good, the waiters were friendly, and the staff was nice. We would hi

Le modèle GPT2 n'étant pas spécialisé pour cette tâche et il ne parvient pas à répondre à notre demande, il génère au contraire des exemples d'avis qui pourrait provenir du même restaurants.

On se tourne alors vers un autre modèle DIALO-GPT.

3.9

LLM Dialo- GPT pour la génération de texte

DialoGPT est un modèle spécialement conçu pour comprendre et générer du texte dans le contexte des dialogues. Il s'agit d'une variante du modèle GPT (*Generative Pretrained Transformer*), entraîné pour **suivre le style et le flux d'une conversation**, ce qui peut être utile pour extraire des insights de manière plus naturelle et conversationnelle à partir de données textuelles comme les avis des clients.

Ici, on définit la fonction *generate_dialogue_insights* : Cette fonction prend une liste d'avis et les combine en un texte (combined_reviews). Elle crée ensuite un prompt qui fournit les avis au modèle et demande d'identifier les faiblesses du restaurant.

```
def generate_dialogue_insights(reviews):
    combined_reviews = " ".join(reviews[:5]) |
    prompt = (
        f"Customer Reviews: {combined_reviews}\n"
        f"AI: Based on these reviews, the key weaknesses of the restaurant are :"
    )

    # Générer une réponse basée sur les avis
    generated_response = dialogue_generator(prompt, max_length=200, num_return_sequences=1)
    return generated_response[0]['generated_text']
```

3.9 LLM Dialo-GPT pour la génération de texte

Exemple: ici les insights générés mettent bien en évidence les points faibles évoqués

Business ID: 123

Insights:

Customer Reviews: Great service but the food was bland. Loved the ambiance and the dessert, but the main course was too salty.

AI: Based on these reviews, the key weaknesses of the restaurant are : 1. Saltiness 2. Saltiness 3. Saltiness 4. Saltiness 5. Saltiness

Le modèle Dialo-GPT semble démontrer une meilleure performance que le modèle GPT2, s'adaptant mieux à cette tâche car il est spécifiquement conçu pour les tâches de dialogue, ce qui lui permet de mieux comprendre et synthétiser les retours clients, qui sont souvent structurés comme une conversation.

4 Application

Sur l'application, nous pouvons trouver un menu déroulant permettant de choisir le restaurant que nous voulons étudier. Le bouton “Afficher un avis au hasard” de ce restaurant permet de générer un avis dans notre dataset, et il affiche ainsi les différentes prédictions testées dans notre étude: la prédiction avec Vader, Bert et CNN.

The screenshot shows a user interface for analyzing restaurant reviews. At the top, there's a logo with a small profile icon and the word "YELP". Below it, a heading reads "L'analyse des avis des restaurants YELP". A dropdown menu labeled "Choisissez un restaurant" contains the option "Le Comptoir de la Gastronomie", which is highlighted with a pink background. Below the dropdown is a button labeled "Afficher un avis au hasard". To the right, under the heading "Avis sélectionné au hasard:", there's a section for "Avis original;" showing a snippet of a review in French: "Based on the menu presented, one could write a glowing review like this Les Classiques offers an exquisite culinary journey into the heart of French...". Below this, under "Prediction de sentiment avec Vader:", the word "positive" is listed. Further down, under "Prediction de sentiment avec Bert:", another "positive" prediction is shown. At the bottom, under "Prediction de sentiment avec CNN:", the word "positive" appears again.

L'analyse des avis des restaurants
YELP

Choisissez un restaurant

Le Comptoir de la Gastronomie

Afficher un avis au hasard

Avis sélectionné au hasard :

Avis original;

Based on the menu presented, one could write a glowing review like this
Les Classiques offers an exquisite culinary journey into the heart of French...

Prediction de sentiment avec Vader:

positive

Prediction de sentiment avec Bert:

positive

Prediction de sentiment avec CNN:

positive

4 Application

Egalement nous pouvons trouver une analyse de sentiment des aspects (ABSA). Il est composé d'un menu déroulant avec quelques phrases d'exemples. Le bouton "Analyser les aspects" permet en sortie d'analyser chaque topic de la phrase en lui attribuant un sentiment.

Analyse de sentiment des aspects (ABSA)

Choisir un commentaire pour l'analyse :

The ambiance was lovely as well as the service, but the food was not that good.

Analyser les aspects

Résultats de l'analyse des aspects :

```
▼ {  
  "span" : "ambiance"  
  "polarity" : "positive"  
}
```

```
▼ {  
  "span" : "service"  
  "polarity" : "positive"  
}
```

```
▼ {  
  "span" : "food"  
  "polarity" : "negative"  
}
```

4 Application

Par ailleurs, nous pouvons résumer les avis des restaurants. Ce volet est composé d'un menu déroulant avec les id des restaurants qu'on peut sélectionner dans notre dataset. La sliding bar permet de choisir le nombre de phrase que comporte le résumé. Le bouton "Générer un résumé" permet en sortie d'afficher un résumé selon les conditions.

Résumé des avis des restaurants

Choisissez un ID de restaurant

-YKc0e0z_47s8JeXA2feOg

Choisissez le nombre de phrases pour le résumé



Générer un résumé

Avis pour le restaurant sélectionné :

Great place to taste the French food . They are a large menu in order to make everyone happy serving was good and friendly thank you Two days into my trip to Paris I started getting worried. The food was generally quite disappointing. This is Paris Every meal should be amazing. I was... Maybe I ordered the wrong dish, but I got steak actually, we all got steak, cooked medium I order medium rare in the US and find that medium in France is...

Résumé :

They are a large menu in order to make everyone happy serving was good and friendly thank you Two days into my trip to Paris I started getting worried. This is Paris Every meal should be amazing.

4 Application

Enfin, nous pouvons trouver un GPT pour le dialogue c'est à dire dans le prompt nous lui demandons quelles sont les faiblesses du restaurant d'après les commentaires pour qu'il puisse nous générer une analyse précise. Ce volet est composé d'un menu déroulant comportant les business ID et en sortie il y concatène tous les commentaires sur ce restaurant. Puis le bouton "Générer des insights" va afficher les points d'améliorations.

DialoGPT Insights Generator

Sélectionnez un restaurant pour générer des insights à partir des avis des clients.

Choisissez un business ID

123

Avis sur le restaurant sélectionné :

The waiter was rude. The steak was fantastic, but the room was too noisy. Amazing cocktails and friendly staff, but the price is too high.

Générer des insights

Insights générés : Customer Reviews: The waiter was rude The steak was fantastic, but the room was too noisy Amazing cocktails and friendly staff, but the price is too high. AI: Based on these reviews, the key weaknesses of the restaurant are: 1. The food is not great 2. The service is not great 3. The service is not great

Nous avons rencontré des problèmes au niveau du déploiement, les fichiers et modèles étant trop lourd.