Université Cadi Ayyad

Faculté des Scienecs Semlalia

Marrakech

Rapport de projet :Sentiment,Émotions , Sarcasm English

Réalisé par : ***EL MASSAFI NISRYNE***

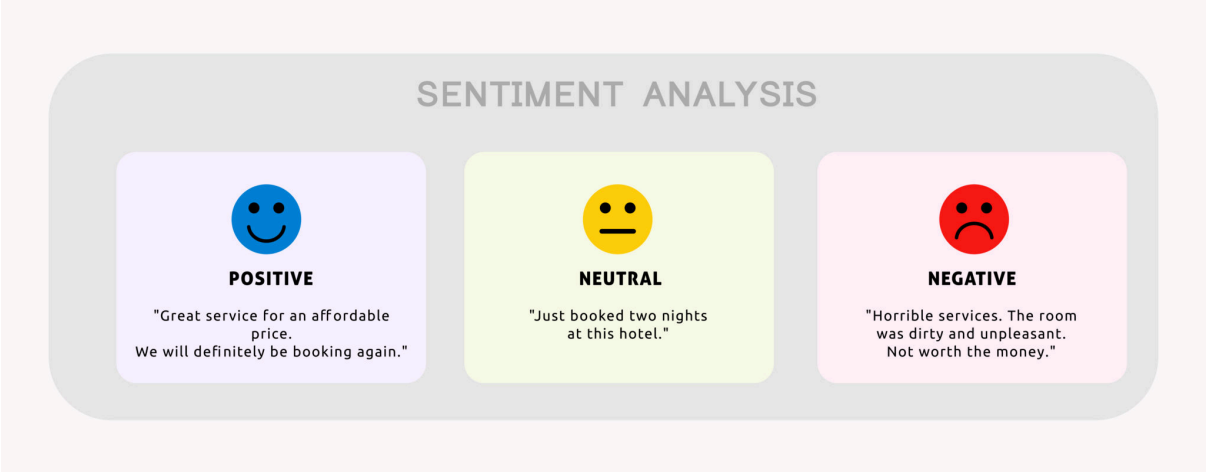
EL BACHA CHAYMA

Demandé par : ***Pr.Jihad Zahir***

I. Sentiment.

0.1 Description

L'analyse de sentiments est un domaine de l'intelligence artificielle et du traitement automatique du langage naturel (TALN) qui vise à identifier et à catégoriser les opinions exprimées dans un texte, en particulier pour déterminer si l'attitude de l'auteur est positive, négative, ou neutre.



0.2 Model utilisée :

On a utilisé dans notre projet deux approche différente :

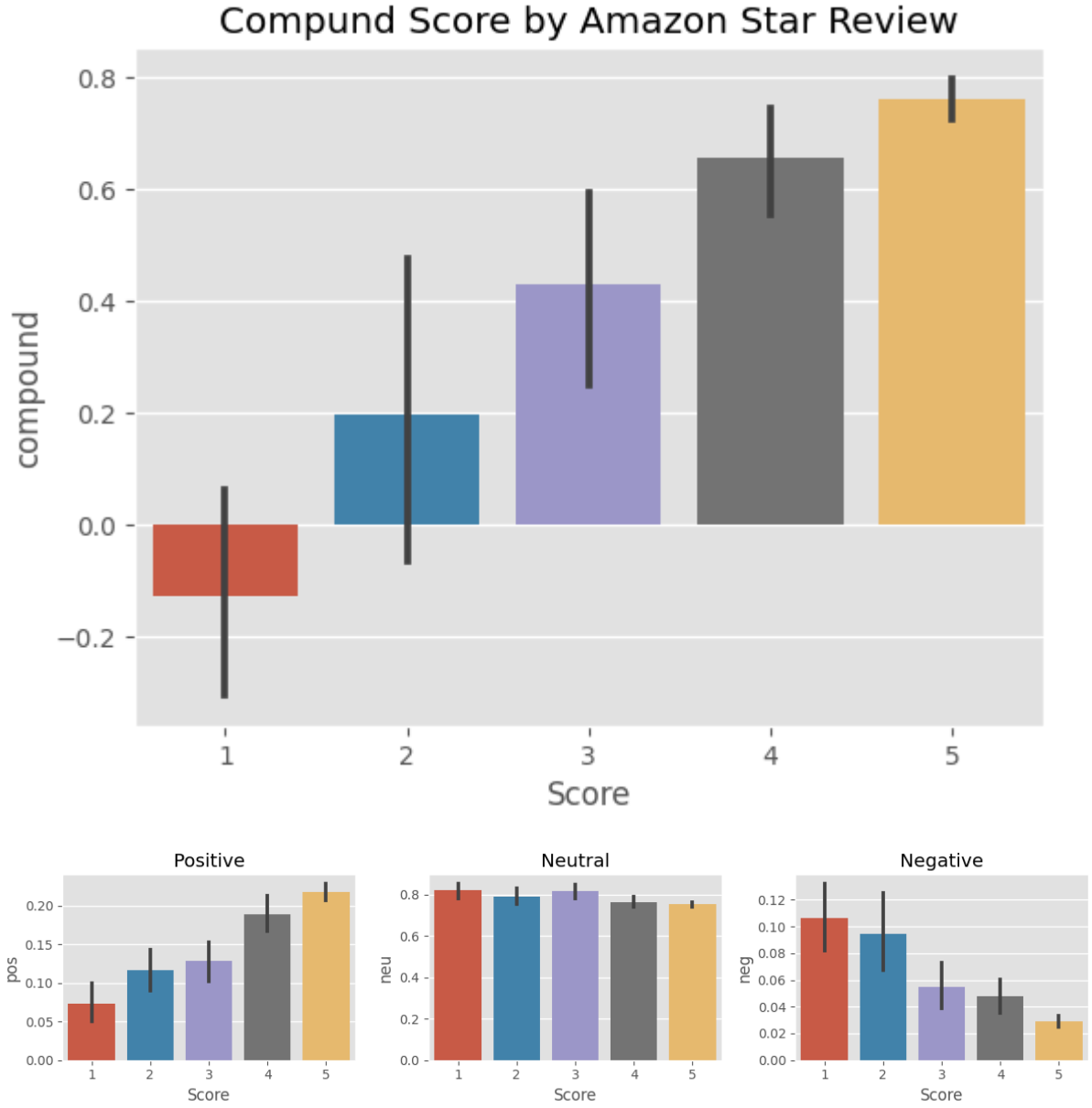
1. **VADER (Valence Aware Dictionary and sentiment Reasoner) - Bag of words approach**

On a utilisé la dataset Reviews.csv Cette base de données contient généralement des avis de consommateurs sur des produits alimentaires vendus sur Amazon.

Étape 1:VADER Sentiment Scoring

Nous utiliserons le SentimentIntensityAnalyzer de NLTK pour obtenir les scores neg/neu/pos du texte.Cela utilise une approche « bag of words » :

Les mots vides sont supprimés, chaque mot est noté et combiné pour obtenir un score total.

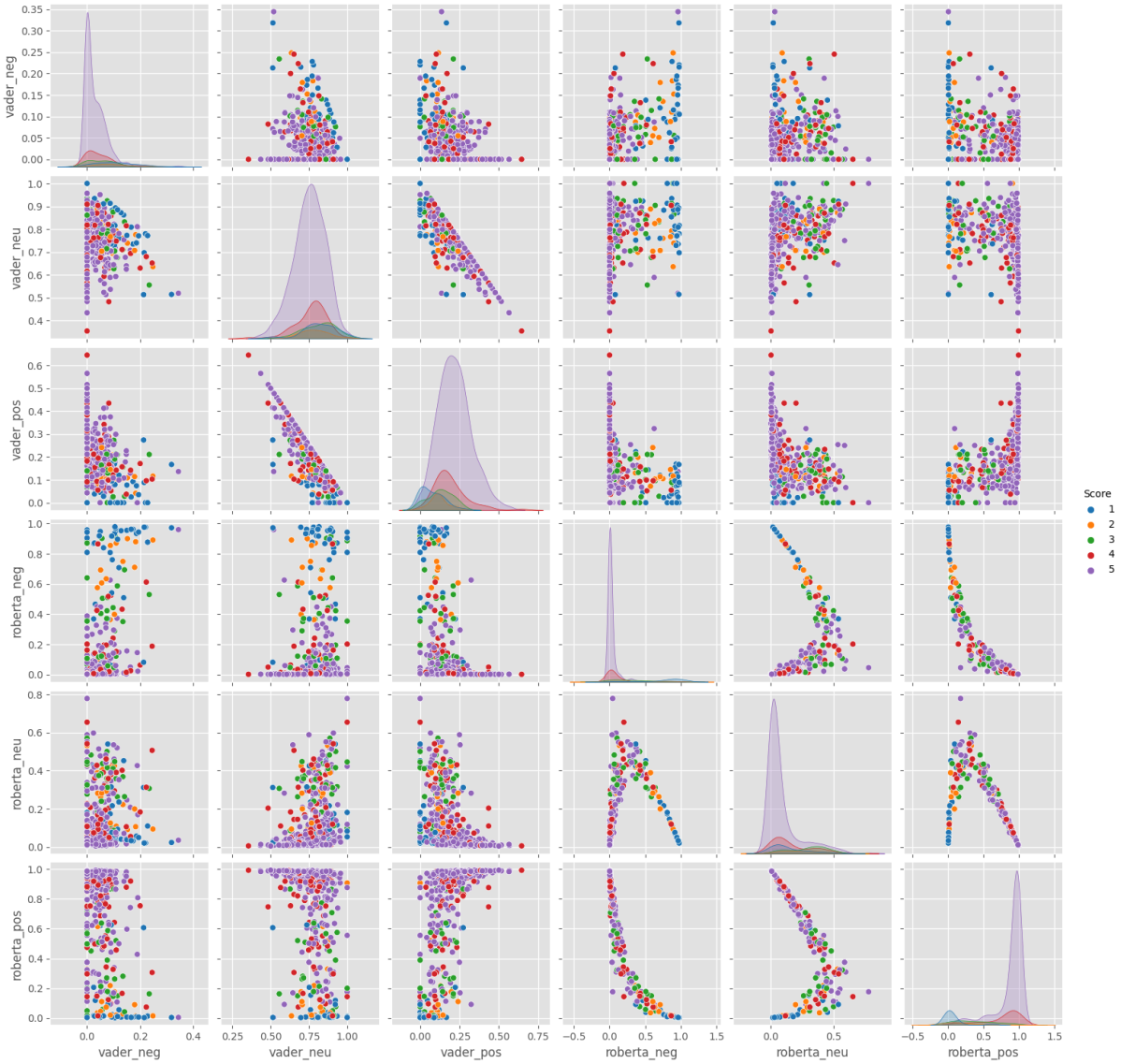


L'image illustre une analyse comparative de la distribution des sentiments classés en trois catégories : Positif, Neutre et Négatif.

Étape 3. Modèle pré-entraîné Roberta

- Utilisez un modèle entraîné à partir d'une grande quantité de données.
- Le modèle Transformers prend en compte les mots mais aussi le contexte lié aux autres mots.

Ici on a Combiner et comparer les deux approches Roberta et VAder



0.3 Résultat:

pour Roberta

```
[ ]  # Run for Roberta Model
     encoded_text = tokenizer(example, return_tensors='pt')
     output = model(**encoded_text)
     scores = output[0][0].detach().numpy()
     scores = softmax(scores)
     scores_dict = {
         'roberta_neg' : scores[0],
         'roberta_neu' : scores[1],
         'roberta_pos' : scores[2]
     }
     print(scores_dict)
```

```
{'roberta_neg': 0.97635514, 'roberta_neu': 0.020687476, 'roberta_pos': 0.002957372}
```

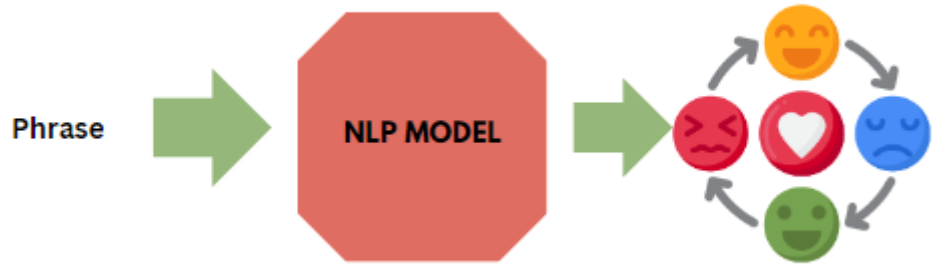
Pour VADER:

```
[ ]  sia.polarity_scores('I am so happy!')

     {'neg': 0.0, 'neu': 0.318, 'pos': 0.682, 'compound': 0.6468}

[ ]  sia.polarity_scores('This is the worst thing ever.')

     {'neg': 0.451, 'neu': 0.549, 'pos': 0.0, 'compound': -0.6249}
```

II. Emotion

0.1 Description

Cette section du projet est consacrée à l'exploration des capacités de détection des émotions en utilisant un modèle pré-entraîné de traitement du langage naturel (NLP). L'objectif est de discerner avec précision les émotions variées et souvent subtiles exprimées à travers le texte. Les émotions, telles que la joie, la tristesse, la colère, la surprise, la peur, et d'autres nuances émotionnelles, jouent un rôle crucial dans la compréhension du contexte et du ton d'un message.



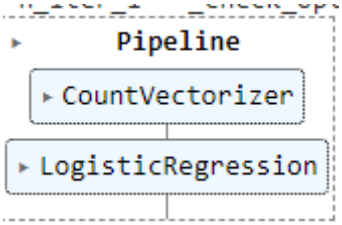
0.2 Model utilisée :

Dans cette phase de notre projet, nous essayons d'utiliser EmoRoBERTa, un modèle de traitement du langage naturel de pointe, pour relever un défi unique : la détection et l'interprétation des émojis dans les textes. Les émojis, omniprésents dans la communication numérique moderne, offrent une fenêtre fascinante sur les émotions et les intentions des utilisateurs. Cependant, leur interprétation précise nécessite une compréhension nuancée du contexte et du ton, souvent difficile à saisir pour les algorithmes standard. EmoRoBERTa, avec sa capacité avancée à analyser les nuances émotionnelles, est idéalement positionné pour décoder ces symboles riches en significations.

```
emotion = pipeline('sentiment-analysis', model='arpanghoshal/EmoRoBERTa')
```

Application de la Régression Logistique pour la Détection des Émotions

et nous avons aussi essayé d'utiliser algorithme de Régression logistique de détection d'émotions à partir de textes, nous avons intégré l'approche de la régression logistique, un modèle classique de machine learning. Cette approche a été choisie pour sa simplicité, sa facilité d'implémentation et sa capacité à traiter efficacement des problèmes de classification multiclasse. tout en conservant les caractères spéciaux pour leur potentiel de signification émotionnelle. Nous avons ensuite préparé notre modèle en construisant une pipeline contenant un transformateur CountVectorizer pour convertir les textes en un format exploitable par le modèle, suivi de l'estimateur LogisticRegression. Cette pipeline a été entraînée sur un ensemble de données d'entraînement.



0.3 l'ensemble de données Emotion

cette dataset se trouve dans kaggle

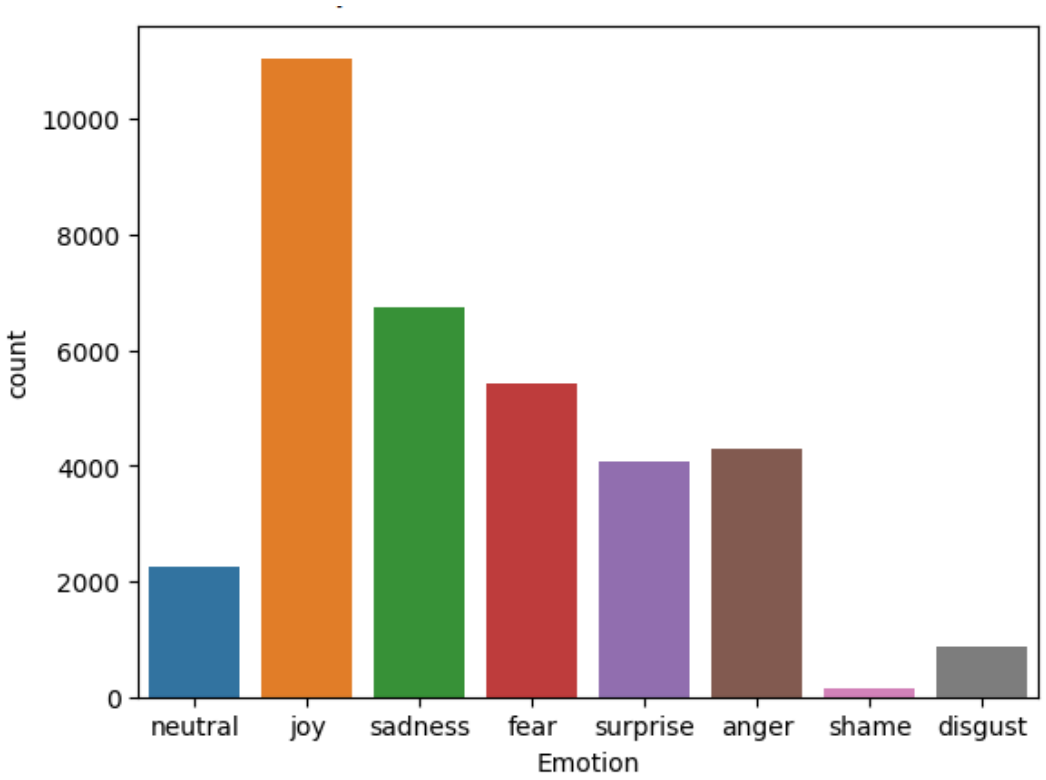
Contenu

Chaque enregistrement se compose de deux attributs :

- **Emotion : joy/sadness**
- **Text :**

Après une phase initiale d'analyse exploratoire des données (EDA) où nous avons examiné la distribution des différentes émotions dans notre jeu de données, nous avons procédé à un nettoyage minutieux des textes, en éliminant les éléments non pertinents tels que les identifiants d'utilisateur et les mots vides,
supprimer stopwords

```
# Stopwords
df['Clean_Text'] = df['Clean_Text'].apply(nfx.remove_stopwords)
```



0.3 Résultat:

pour Logistic Regression

```
# Make A Prediction
ex1 = "This book was so interesting it made me happy"
```

```
pipe_lr.predict([ex1])
```

```
array(['joy'], dtype=object)
```

pour EmoRoBERTa,

```
# Phrase à tester
test_phrase = "terrible  furious  impotent anger filled  making want strike  shout scream "

# Utiliser la pipeline EmoRoBERTa pour prédire l'émotion
predicted_emotion = emotion(test_phrase)
print(predicted_emotion)
```

```
[{'label': 'anger', 'score': 0.9863088726997375}]
```

❖ conclusion

Après plusieurs itérations et évaluations de différents modèles, nous avons choisi de déployer l'algorithme de régression logistique pour cette tâche. Ce choix a été guidé par plusieurs facteurs clés. Tout d'abord, la régression logistique a démontré une capacité remarquable à gérer la classification, une exigence cruciale étant donné la variété des émojis et des émotions qu'ils représentent. De plus, sa simplicité et son efficacité en termes de calcul en font une solution idéale pour une intégration rapide et fiable dans des systèmes en temps réel.

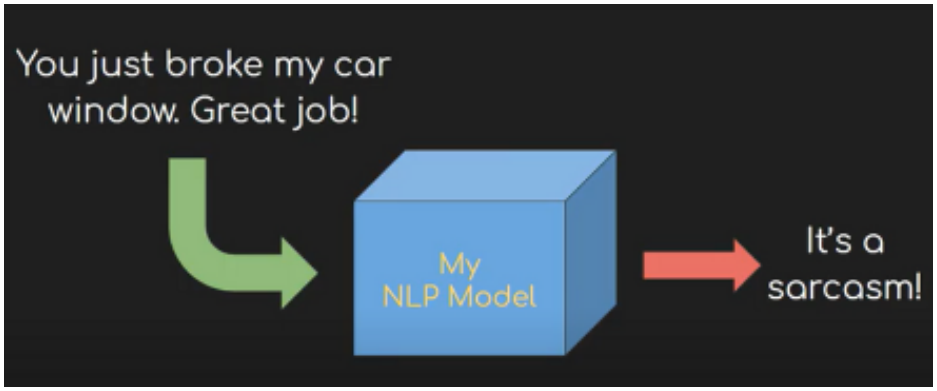
III. Sarcasm

01. Description

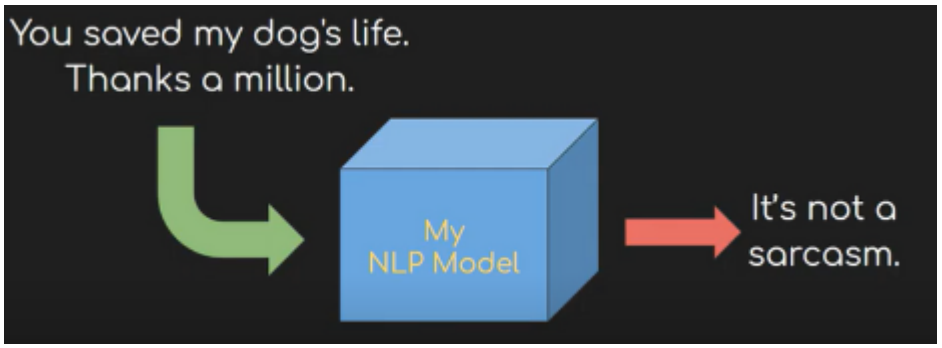
Cette partie de projet vise à explorer les capacités de détection de sarcasme à l'aide d'un modèle de traitement du langage naturel (NLP) pré-entraîné. Le sarcasme, caractérisé par une divergence entre le sens littéral et le sens implicite des propos, représente un défi majeur dans l'analyse de sentiments et la compréhension contextuelle.

voici un schéma qui représente le démarrage de cette partie du projet à l'aide de deux exemples

Exemple 1:



Exemple 2:



02. Model pré-entraînée utilisée :

Dans la phase de construction du modèle, nous avons intégré une approche hybride combinant les embeddings de mots pré-entraînés GloVe avec un réseau neuronal récurrent de type LSTM (Long Short-Term Memory).

Le modèle GloVe est couramment utilisé dans le traitement du langage naturel pour convertir des mots en représentations vectorielles capables de capturer la signification sémantique des mots.
Cela signifie que des mots similaires seront placés ensemble.
GloVe est un modèle d'embedding de mots qui encode la sémantique des mots en vecteurs denses, en se basant sur des statistiques globales du corpus.

```
GLOVE_DIR = '/content'
f = open(os.path.join(GLOVE_DIR, 'glove.twitter.27B.100d_wv.txt'), encoding = "utf-8")
```

Ensuite, nous avons utilisé **une couche LSTM**, un type de RNN particulièrement efficace pour traiter les séquences de données en tenant compte de leur contexte temporel. Les LSTMs sont capables de capturer la dépendance à long terme et sont donc particulièrement adaptés à la compréhension du sarcasme, qui nécessite souvent une analyse contextuelle des phrases.

```
model = Sequential()
model.add(embedding_layer)
model.add(LSTM(64, dropout=0.2, recurrent_dropout=0.25))
```

Cette combinaison de GloVe et de LSTM vise à exploiter à la fois les représentations de mots riches en sémantique et la capacité des LSTMs à comprendre la structure séquentielle et le contexte du langage, éléments clés pour identifier avec précision le sarcasme dans le texte.

03.l'ensemble de données sarcasm

ensembles de données Twitter collectés à l'aide d'une supervision basée sur les hashtags,
Cet **ensemble de données News Headlines pour la détection du sarcasme** est collecté à partir de deux sites Web d'actualités. TheOnion vise à produire des versions sarcastiques de l'actualité et collecte tous les titres des catégories News in Brief et News in Photos (qui sont sarcastiques). collecte des titres d'actualité réels (et non sarcastiques) du *HuffPost* .

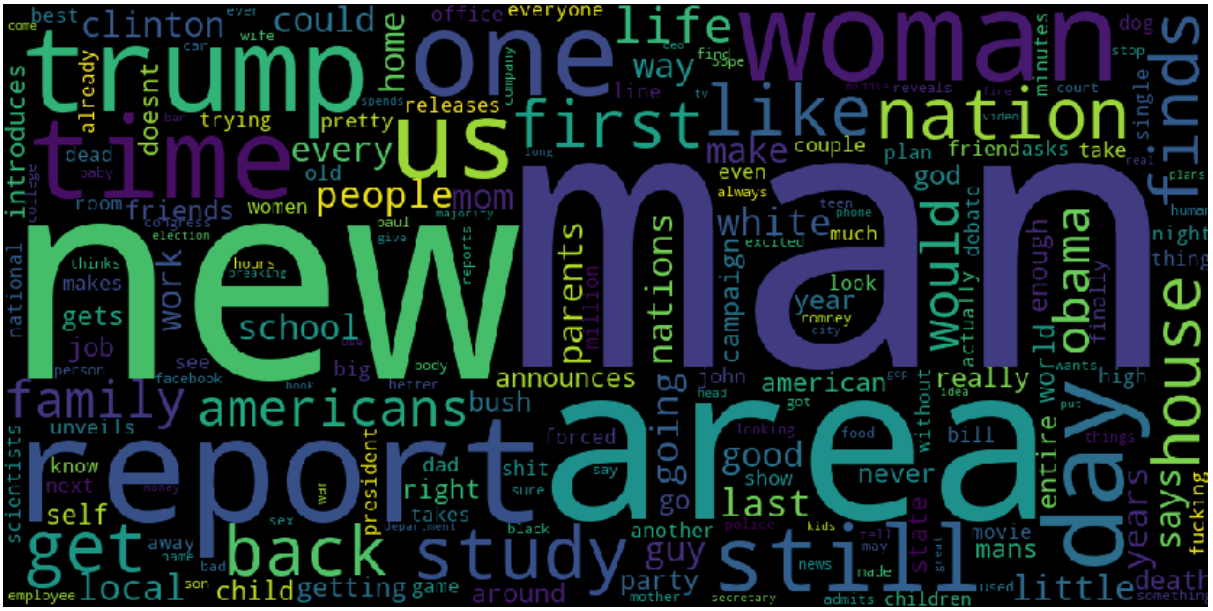
Contenu

Chaque enregistrement se compose de trois attributs :

- `is_sarcastic`: 1 si la notice est sarcastique sinon 0
- `headline`: le texte
- `article_link`: lien vers l'article d'actualité original. Utile pour collecter des données supplémentaires

	is_sarcastic	headline	article_link
0	1	thirtysomething scientists unveil doomsday clo...	https://www.theonion.com/thirtysomething-scien...
1	0	dem rep. totally nails why congress is falling...	https://www.huffingtonpost.com/entry/donna-edw...
2	0	eat your veggies: 9 deliciously different recipes	https://www.huffingtonpost.com/entry/eat-your-...
3	1	inclement weather prevents liar from getting t...	https://local.theonion.com/inclement-weather-p...
4	1	mother comes pretty close to using word 'strea...	https://www.theonion.com/mother-comes-pretty-c...

Visualisation les mots les plus fréquents:



04. Résultat:

ex01:

```
predict_sarcasm("I was depressed. He asked me to be happy. I am not depressed anymore.")
```

```
"It's a sarcasm!"
```

ex02

```
predict_sarcasm("You just broke my car window. Great job.")
```

```
"It's a sarcasm!"
```

ex03

```
predict_sarcasm("You just saved my dog's life. Thanks a million.")
```

```
"It's not a sarcasm."
```

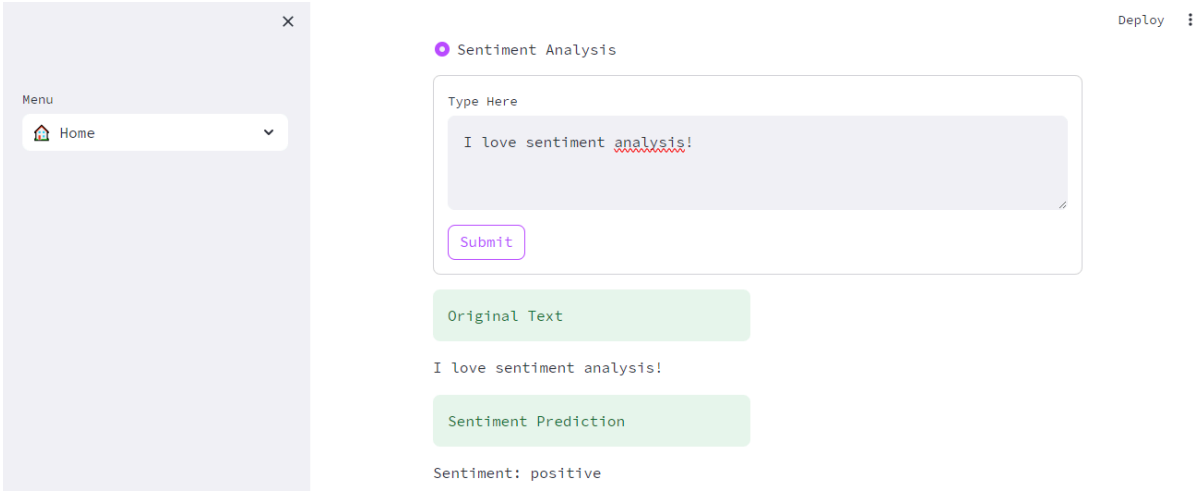
- ❖ **conclusion**

En conclusion, ce modèle hybride GloVe-LSTM a démontré une capacité remarquable à différencier efficacement entre les expressions sarcastiques et non sarcastiques, prouvant ainsi son utilité et sa précision dans la détection nuancée du sarcasme dans le langage naturel.

IV. Deployement

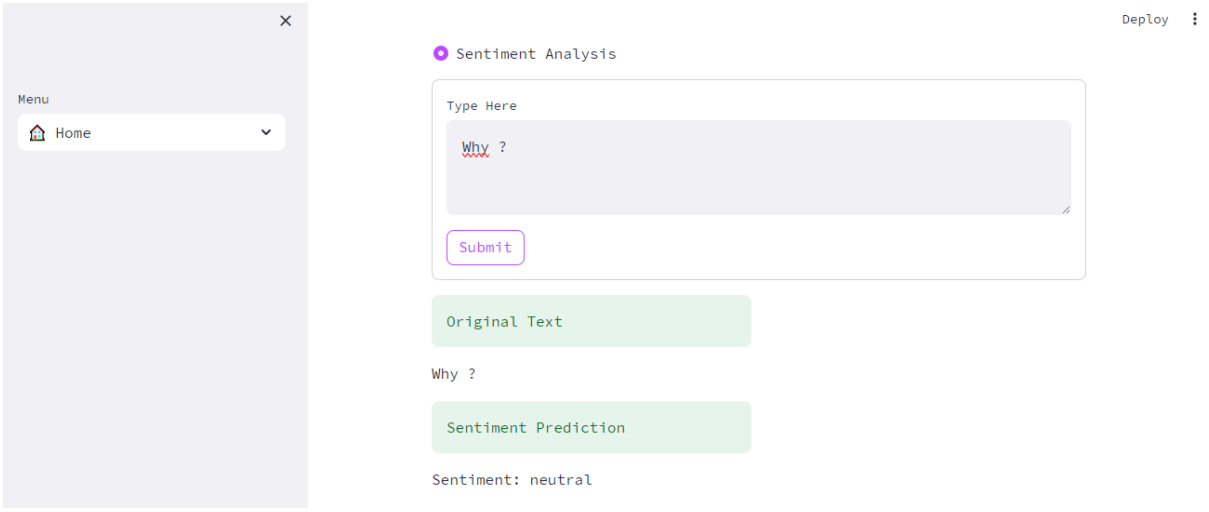
- ***Analyse de sentiments***

positive sentiment 😀

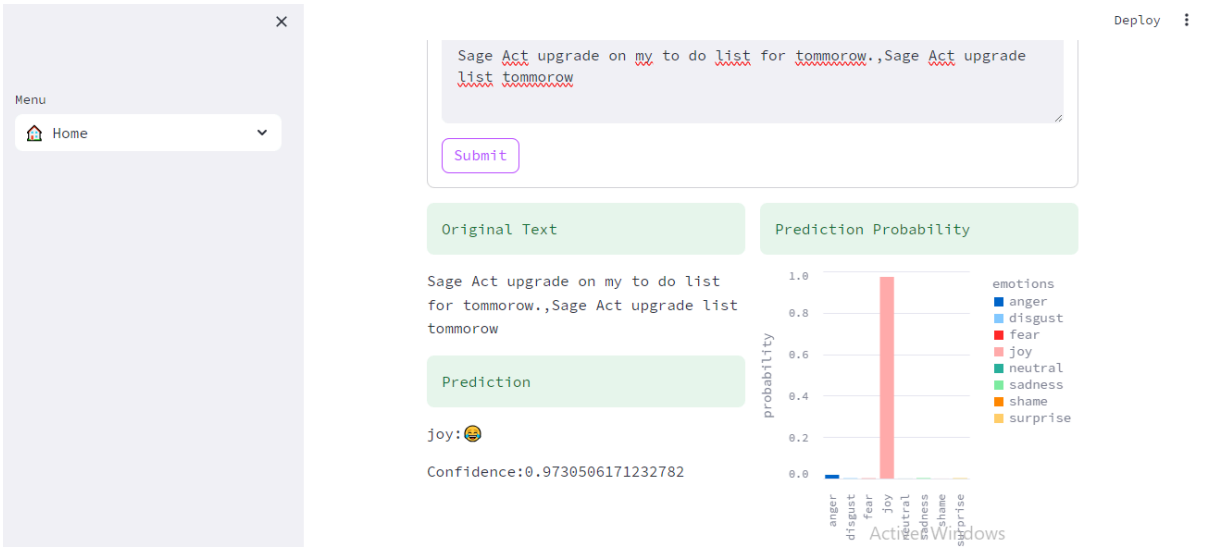


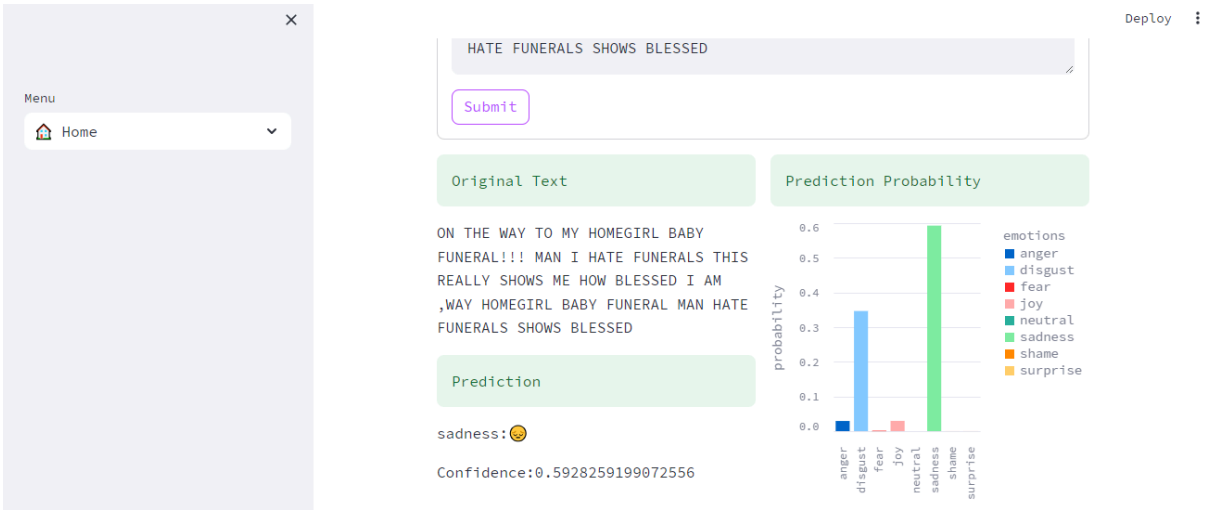
negative sentiment 😡

sentiment neutre 👍

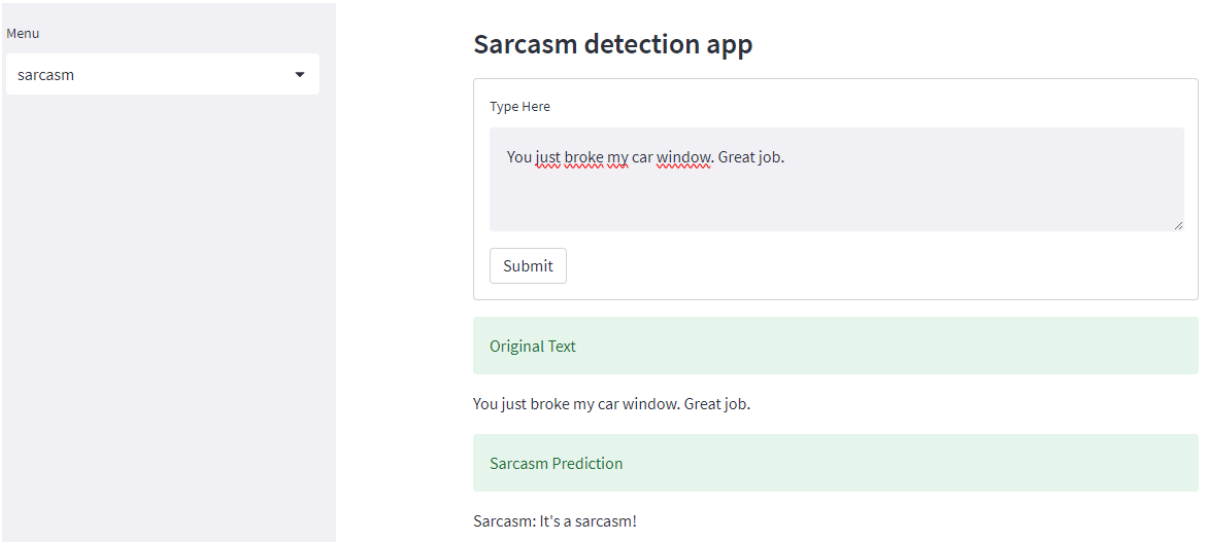


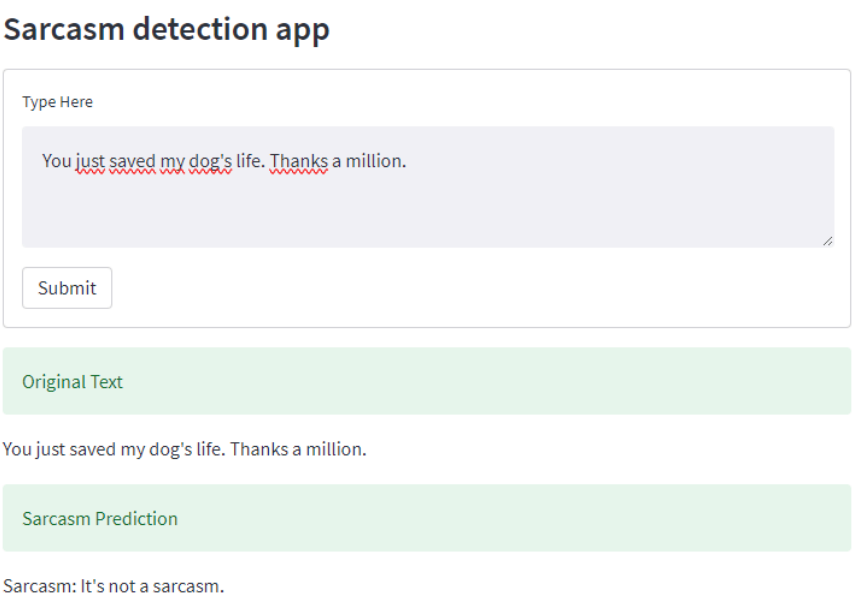
- ***Détection d'émotion***





- ***Sarcasm détection***


Menu
sarcasm
Sarcasm detection app
Type Here
You just broke my car window. Great job.
Submit
Original Text
You just broke my car window. Great job.
Sarcasm Prediction
Sarcasm: It's a sarcasm!


Sarcasm detection app
Type Here
You just saved my dog's life. Thanks a million.
Submit
Original Text
You just saved my dog's life. Thanks a million.
Sarcasm Prediction
Sarcasm: It's not a sarcasm.