Solution

1) Pipeline de traitement processus de détection d'un langage abusif sur le Social Web :

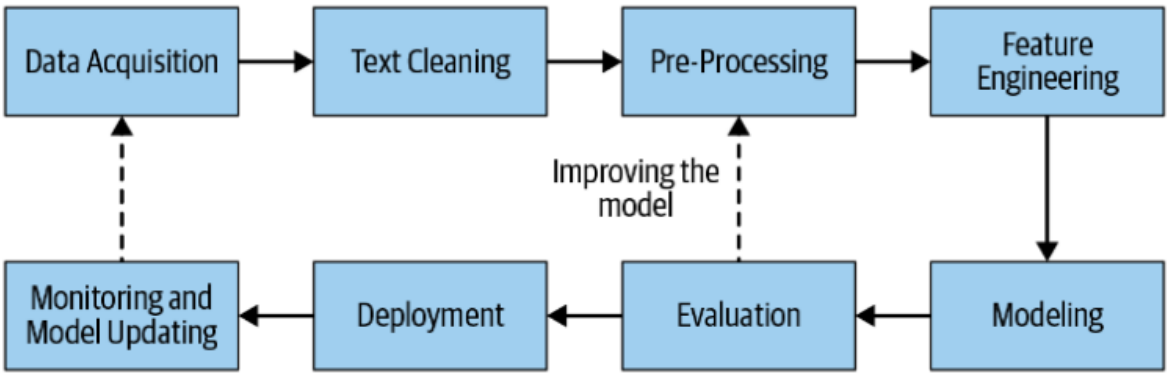

Figure 1: Pipline NLP

Text Acquisition : collection des données pertinentes c-à-d rassembler presque tous données textuelle abusifs (commentaires, textes, ……) après les annoter en plusieurs classes par exemple (toxique, obscène, insulte…)

Text Cleaning :

-Tokenisation et Capitalisation / Décapitalisation

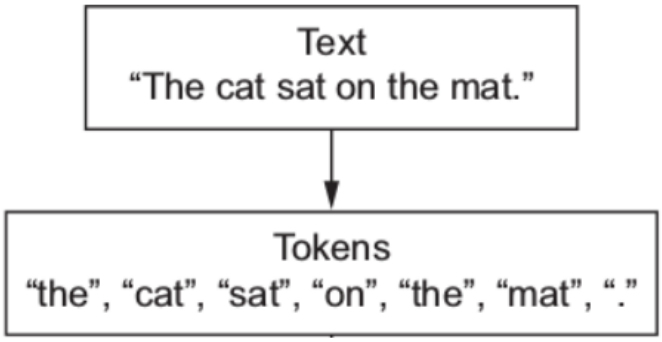

Figure 2: Tokenisation d'une phrase.

-Suppression des stops words.

-Suppression des caractères indésirables.

-Encodage dans le bon format (l'encodage UTF-8, l'encodage latin, les encodages ISO / CEI, etc.

- Rompre les mots attachés

Pre-Processing : mettre le texte sous une forme prévisible **et** analysable.

-Soit par la vectorisation ou l'utilisation des sacs de mots.

- Lemmatisation / Stemming

-Utilisation des techniques de TF-IDF, des mesures de distances (cos, distance euclidienne,)

...

Feature Engineering : Le choix des variables qui signifie bien notre cible.

Modeling : La partie machine learning et de choisir les algorithmes convenables.

Evaluation :

- Evaluer les modèles construits à partir des métriques convenables au problème (par exemple log loss)

- Tester le modèle sur le maximum possible des textes pour le prouver.

Après le choix du modèle il faut bien utiliser une version 0 après de rectifier les erreurs après chaque bug.

Pour l'utilisation des réseaux de neurones sur des textes on se concentre juste une technique de word embeding sans faire d'extractions des features.

Remarque :

Il y a l'équipe Conversation AI qui est une initiative de recherche fondée par Jigsaw et Google (tous deux faisant partie d'Alphabet), travaille sur des outils pour aider à améliorer la

conversation en ligne. Un domaine d'intérêt est l'étude des comportements négatifs en ligne, comme les commentaires toxiques (c'est-à-dire les commentaires impolis, irrespectueux ou susceptibles de pousser quelqu'un à quitter une discussion). Jusqu'à présent, ils ont construit une gamme de modèles accessibles y compris la toxicité. Mais les modèles actuels font toujours des erreurs, et ils ne permettent pas aux utilisateurs de sélectionner les types de toxicité qu'ils souhaitent trouver (par exemple, certaines plates-formes peuvent convenir à des blasphèmes, mais pas à d'autres types de contenu toxique).

2) CNN-LSTM pour détection d'un langage abusif sur le Social Web :

L'utilisation des CNN sur des textes se base sur la notion des words/ character embeddings c'est-à-dire de construire une matrice des vecteurs des words/character embeddings après appliquer des noyaux pour avoir des convolutions comme le cas dans les images.

C-LSTM (ou CNN-LSTM) utilise CNN pour extraire une séquence de représentations de phrases de niveau supérieur, et sont introduites dans LSTM à obtenir la représentation de la phrase. C-LSTM est capable de saisir à la fois les caractéristiques locales des phrases ainsi que la sémantique des phrases globales et temporelles.

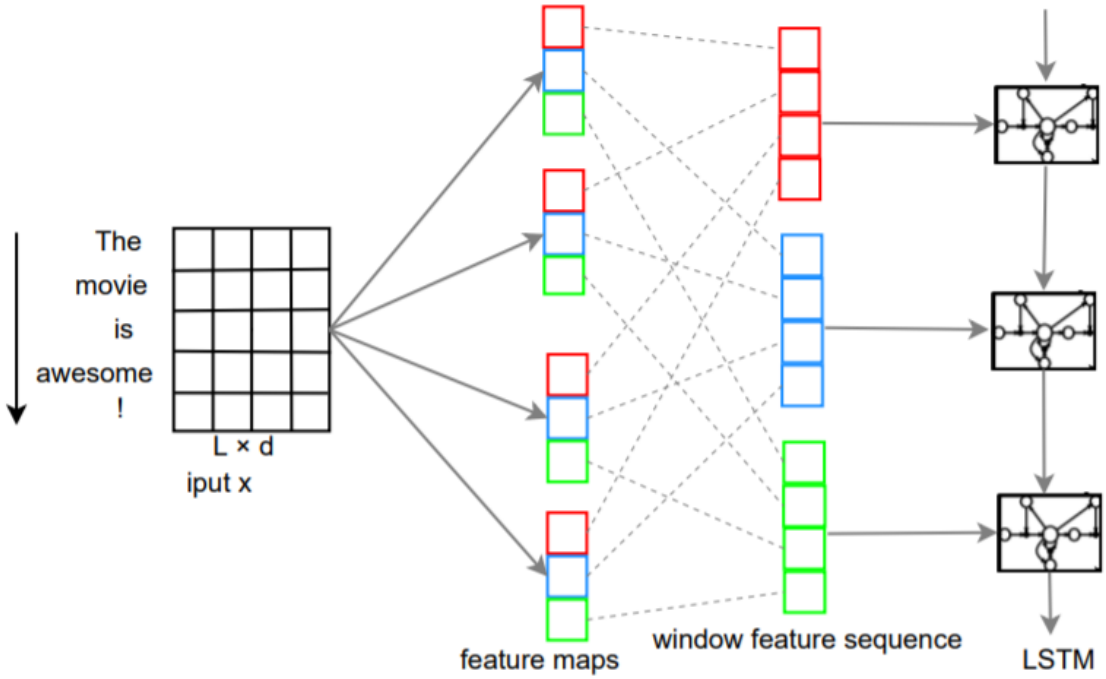

Figure 3: Architecture de C-LSTM

L'architecture du modèle C-LSTM est présentée à la figure 3, qui comprend deux composantes principales : le réseau neuronal convolutif (CNN) et le réseau de mémoire à long terme à court terme (LSTM). Les deux sous-sections suivantes décrivent comment nous appliquons CNN pour extraire des séquences de mots de plus haut niveau et LSTM pour capturer des dépendances à long terme sur des séquences de fenêtres respectivement.

D’après la figure 3, les blocs de même couleur dans feature map et window feature sequence correspond aux caractéristiques de la même fenêtre. Les lignes en pointillés relient l'élément d'une window avec la source de fearture map. La sortie finale du modèle entier est la dernière unité cachée de LSTM.

L'architecture LSTM comporte une série de modules répétés pour chaque pas de temps comme dans un RNN standard. À chaque pas de temps, la sortie du module est contrôlée par un ensemble de portes en fonction de l'ancien état caché et de l'entrée au pas de temps actuel : la

porte d'oubli, la porte d'entrée, et la porte de sortie. Ces portes décident collectivement comment mettre à jour la cellule mémoire actuelle et l'état caché actuel.

3) Collection des commentaires sur le sujet du décès du président Jacques Chirac sur une database MongoDB :

Langage : Python

Bibliothèque :

- **Pymongo** : une distribution Python contenant des outils pour travailler avec MongoDB
- **Selenium**

Voila le code de la création de la databse MongoDB Jaque_Chirac, et les collections postes et commentaires :

```
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
print("connected")
mydb = myclient["Jaque_Chirac"]
posts = mydb["posts"]
comments = mydb["comments"]
post = { "_id" : None, "page_name" : None, "page_url" : None, "post" : None }
comment = { "_id" : None, "page_name" : None , "page_url" : None , "commenter_id" : None , "comment" : None }
```

Figure 4: Le poste à récupérer

Vous trouverez le code en pièce joint.

Résultat :

Après l'execution du code **mongodb. Ipynb** on trouve dans MongoDB Comapss la data base Jaque_Chirac présentée dans la figure 5.

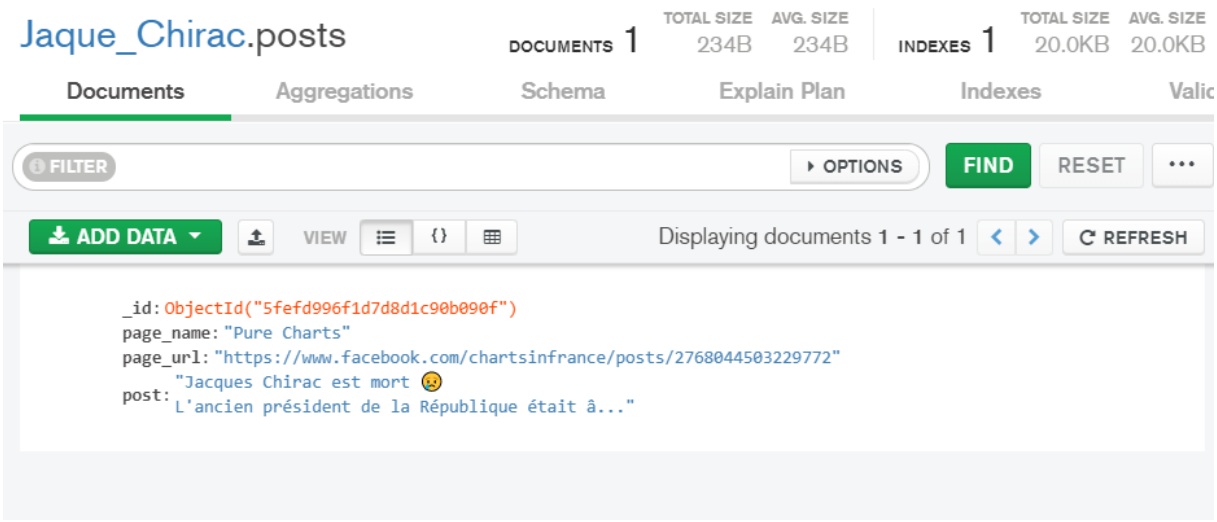

Figure 5: Jaque_Chirac database

Bibliographie

[1] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, Francis C.M. Lau. A C-LSTM Neural Network for Text Classification. *[Submitted on 27 Nov 2015 (v1), last revised 30 Nov 2015 (this version, v2)]*