

The background is a dark blue gradient with abstract white and light blue circular patterns. On the left side, there is a large circular scale with tick marks and numbers ranging from 140 to 260. Several concentric circles and arcs are scattered across the slide, some with arrows indicating a clockwise direction.

# CLASSIFICATION DE TAGS STACK OVERFLOW

ARNAUD FRANCOISE

THIBAUD DIROU

THIBAUT DELATTRE

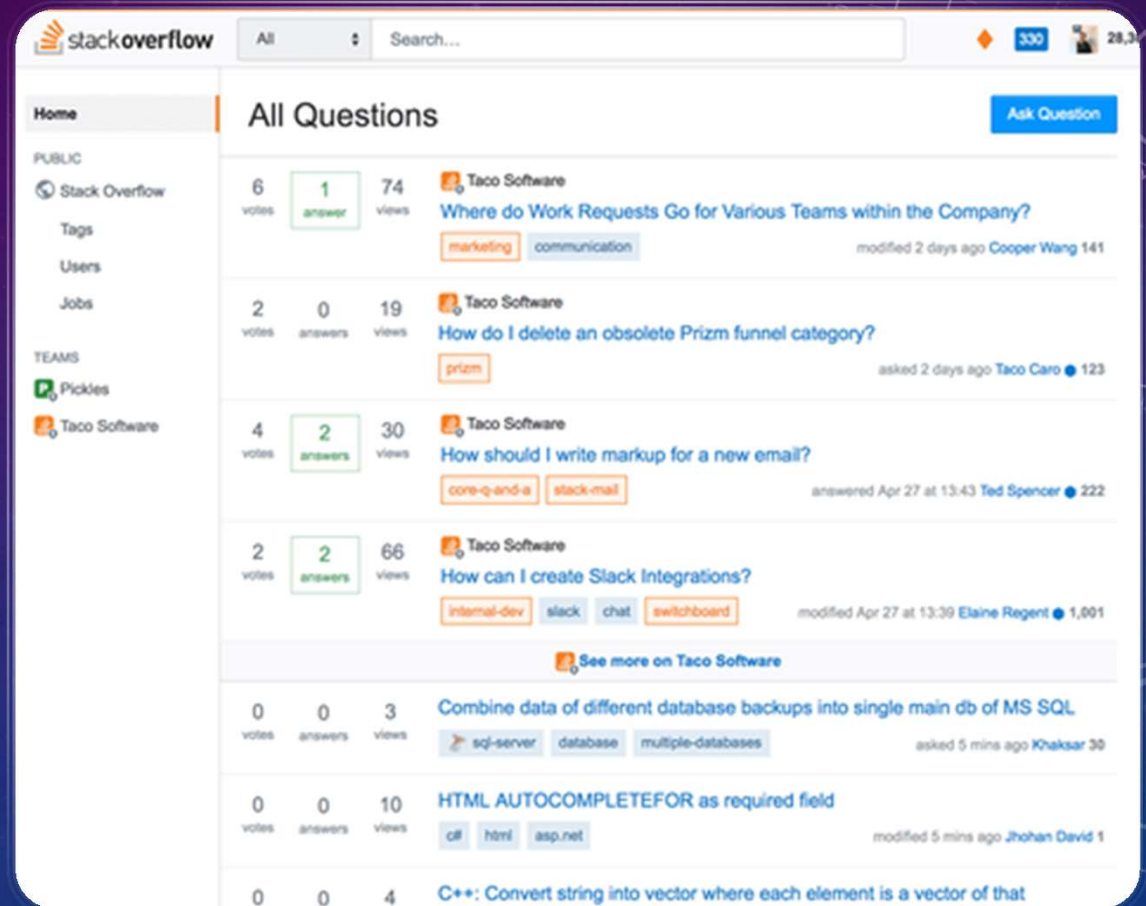
CYRIL DELANNOY

# SOMMAIRE

- Objectif du projet
- Présentation et extraction des données
- Préprocessing
- Modèle (caractéristiques, entraînement)
- Problèmes rencontrés et axes d'amélioration (autres pistes qu'il serait possible d'explorer)

# OBJECTIF

- Prédire les tags associés à une question sur le forum Stack Overflow



# PRÉSENTATION DES DONNÉES

## Qualité :

- Requêtes SQL permettant d'extraire des .csv
- Plusieurs tags par post dans la plupart des posts
- Les posts contiennent des balises html



# EXTRACTION DES DONNÉES

- Choix de 26 tags principaux (Javascript, Python, Java, ...)
- Extraction d'un fichier csv par tag contenant chacun 10000 posts Stack Overflow



# ETAPES DE PREPROCESSING

- Suppression de la ponctuation
- Suppression des stopwords[i have pet]
- Tokenization (transformation des données sous forme de vecteurs) [I, have, a, pet]
- Lemmatization ou Stemming having -> have -> hav

# TF-IDF

TF permet de mesurer l'importance relative d'un mot dans un document.

$$TF(i,j) = \frac{\log_2(1+Freq(i,j))}{\log_2(L_j)}$$

IDF mesure la signification d'un terme non pas en fonction de sa fréquence dans un document particulier, mais en fonction de sa distribution et de son utilisation dans l'ensemble des documents.

$$IDF(i) = \log\left(\frac{N_D}{f_i} + 1\right)$$

Term Frequency – Inverse Document Frequency est une mesure qui permet, à partir d'un ensemble de textes, de connaître l'importance relative de chaque mot.

$$TF - IDF(i,j) = TF(i,j) * IDF(i).$$

# WORD TO SEQUENCE

```
sentences = ['The cat is on the table', 'The dog is running',...]
```



```
tokenizer.fit_on_texts(sentences)
```



```
{'the': 1, 'cat': 2, 'is': 3, 'on': 4, 'table': 5, 'dog': 6, 'running': 7,...}
```

Transforme chaque mot du texte par sa valeur associée dans le dictionnaire

Cette méthode crée un dictionnaire dans lequel chaque mot est associé à un entier.

Plus l'entier associé est petit plus le mot est fréquent

```
Text = ['The dog is running after the cat']
```



```
tokenizer.texts_to_sequences(Text)
```



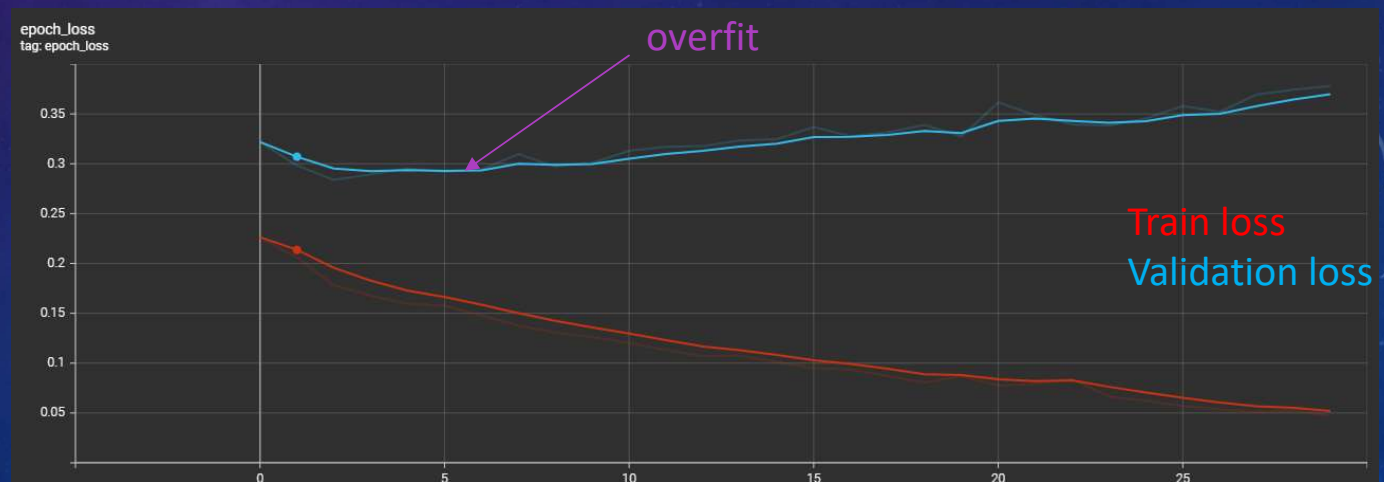
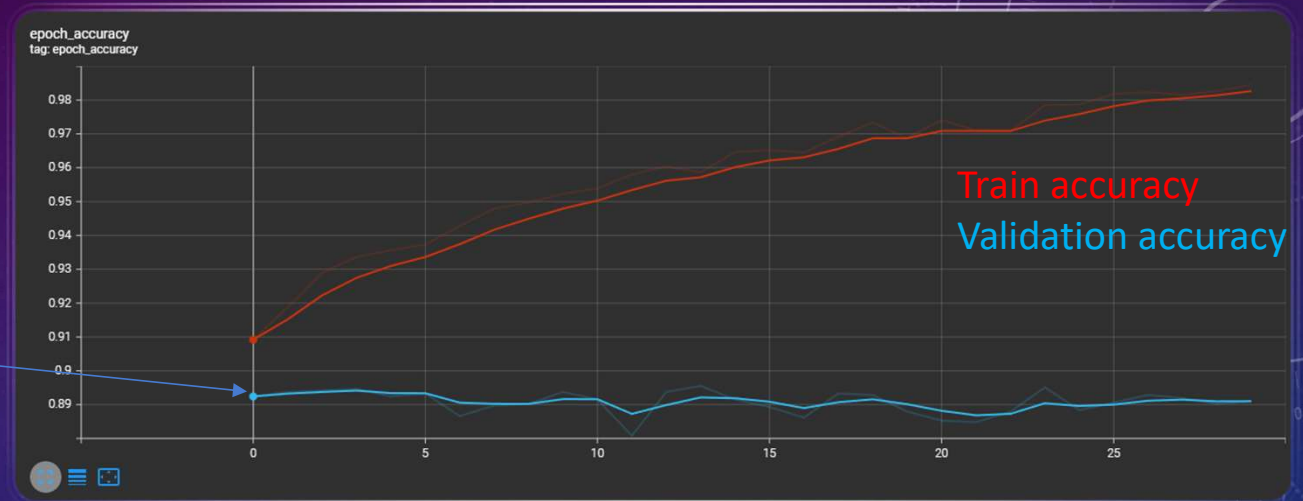
```
[[ 1, 6, 2, 7, 1, 3]]
```



# RNN

89% validation accuracy

- Data:
  - 100 corpus par tags
  - Top 10 tags stackoverflow



# PREDICTIONS

Note pour Mr Dubreu

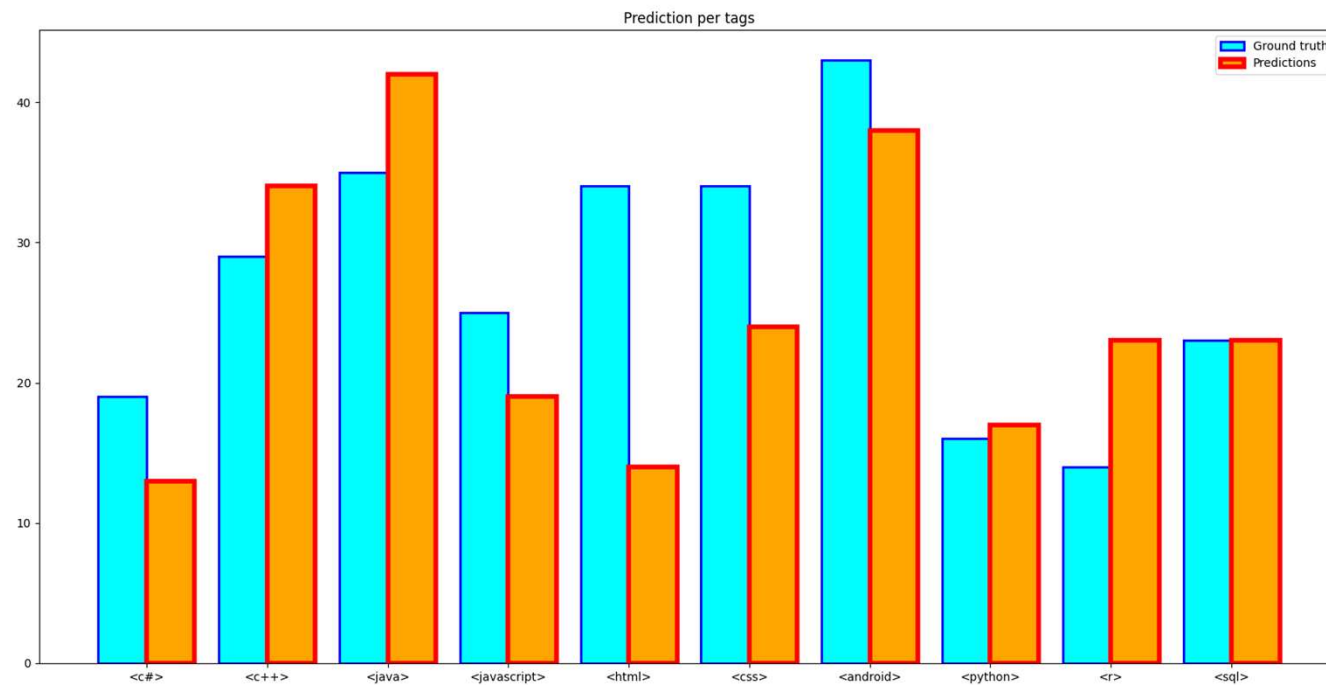
Ici il faut que je refasse les graphiques.

J'ai fais l'erreur de ne pas sauver ma matrice `y_test` qui contient les données inconnues au model désolé.

En attendant de refaire un entrainement en local (4h),

J'ai tracé ses graphiques sur un mix de data test et train (déjà vu)

Si on effectue les mêmes tracé sur les data UNIQUEMENT de TEST on obtient 74% d'accuracy



# PREDICTIONS

Note pour Mr Dubreu

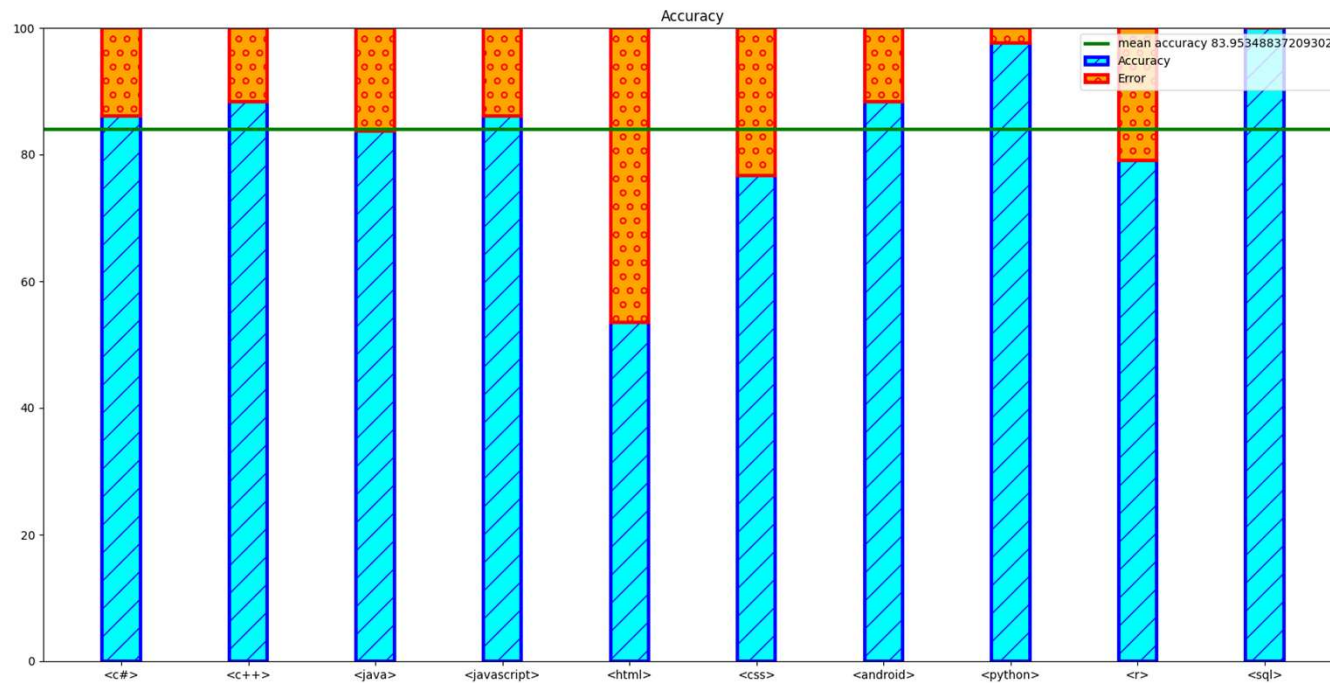
Ici il faut que je refasse les graphiques.

J'ai fais l'erreur de ne pas sauver ma matrice `y_test` qui contient les données inconnues au model désolé.

En attendant de refaire un entrainement en local (4h),

J'ai tracé ses graphiques sur un mix de data test et train (déjà vu)

Si on effectue les mêmes tracé sur les data UNIQUEMENT de TEST on obtient 74% d'accuracy



# PROBLÈMES RENCONTRÉS :

- Taille du dataset trop grand.
- Entraînement du RNN très long.
- Colab limité



# AXES D'AMÉLIORATION

- Augmentation du dataset
- Augmenter le nombre de tags à prédire
- Nécessite une puissance de calcul conséquente