

Parcours Ingénieur Machine Learning

Session Mars 2021

OPENCLASSROOMS

Projet 7

Développez une preuve de concept

23/10/2021

Etudiante : QITOUT Kenza

Mentor et Evalueur : Maïeul Lombard

CONTEXTE DU PROJET

Réaliser une veille thématique sur un problème de Data Science

Problématique : Etude de l'utilisation du Word Embeddings et du Deep Learning lors d'un problème de classification de données textuelles

Objectif : Trouver une méthode plus récente pour améliorer la méthode utilisée en production chez un client

Missions :

- ❖ Implémenter une méthode baseline et la comparer avec une nouvelle méthode
- ❖ Rédiger un Proof Of Concept (POC) présentant l'état de l'art, les évolutions de la Data Science, les avantages, les inconvénients de la nouvelle méthode implémentée

1. EVOLUTION DE LA CLASSIFICATION DES DONNEES TEXTUELLES

Représentation des mots : tf-idf

Fréquence d'apparition des mots dans chaque document en fonction de leur apparition dans les autres documents

Poids = fréquence du n-gram \times idf(n-gramme) \longrightarrow Pondération par un indicateur de similarité $\log\left(\frac{N}{df_i}\right)$

Avantage et inconvénient :

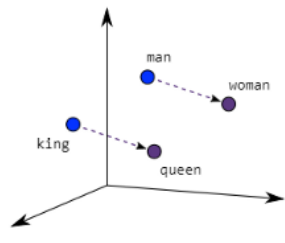
- Méthode simple qui permet de tenir compte si mot rare ou commun dans l'ensemble des documents
- MAIS création de matrices larges et creuses (majorité de 0 dans la matrice)

1. EVOLUTION DE LA CLASSIFICATION DES DONNEES TEXTUELLES

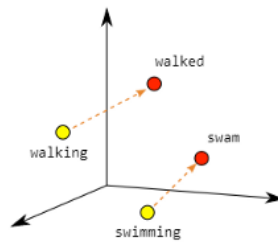
Représentation des mots : Word Embeddings

Récente famille de technique beaucoup utilisée dans le NLP

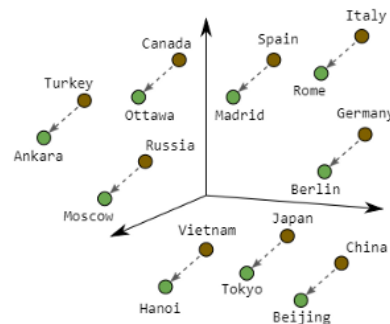
Position d'un mot dans l'espace vectoriel apprise à partir du texte et basée sur les mots qui l'entourent (représentation vectorielle dense)



Male-Female



Verb Tense

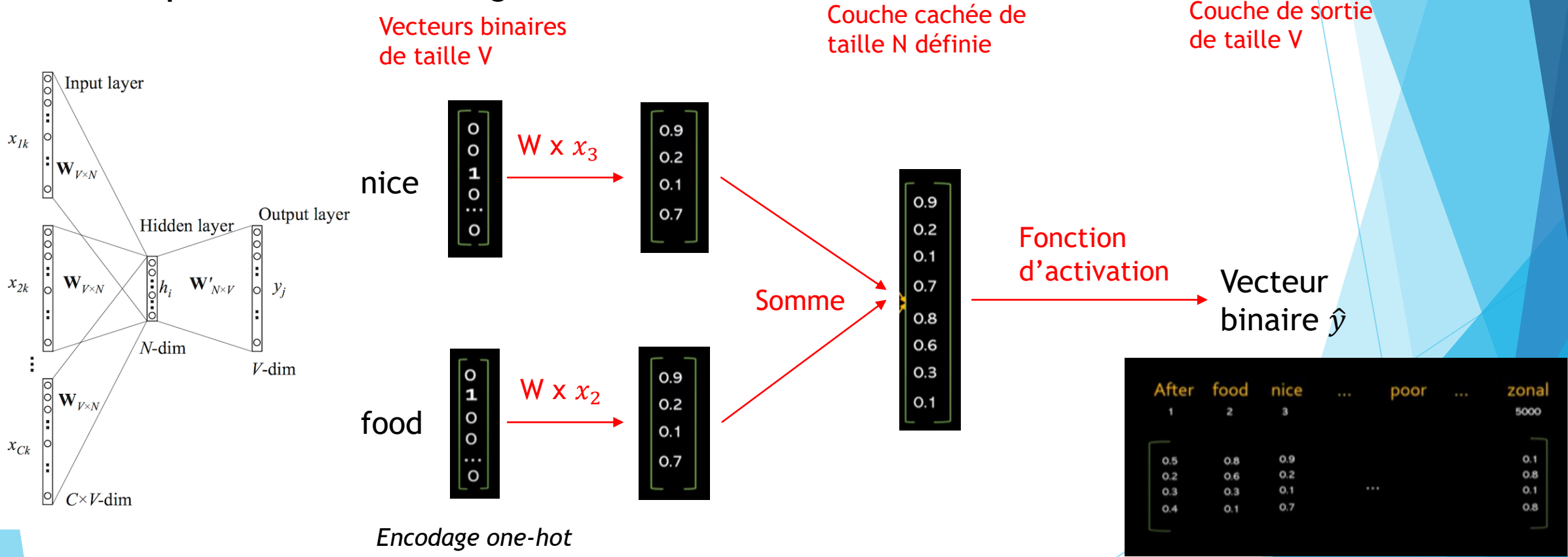


Country-Capital

Mots de même sens
avec représentations
similaires

1. EVOLUTION DE LA CLASSIFICATION DES DONNEES TEXTUELLES

Principe du Word Embeddings :



Prédire un mot en fonction de son contexte (modèle CBOW)

Matrice W de taille $V \times N$ contenant les plongements de mots

1. EVOLUTION DE LA CLASSIFICATION DES DONNEES TEXTUELLES

Représentation des mots : Word Embeddings

After	food	nice	good	poor	weak	...	zonal
							5000
0.5	4.3	0.4	0.38	2.3	2.2		9.8
1.2	0.1	8.1	8.2	1.1	1.0		0.6
0.7	0.9	8.8	8.9	6.7	6.5		2.2
3.1	5.5	4.2	4.1	2.0	1.9		1.3

Matrice W contenant les valeurs des vecteurs de chaque mot

Représentation similaire entre des mots utilisés de la même façon

Avantages :

- Capturer le sens des mots en tenant compte de l'environnement de chaque mot
- Réduire la taille du vecteur final et donc de la représentation des documents
- Pas le problème des matrices creuses

1. EVOLUTION DE LA CLASSIFICATION DES DONNEES TEXTUELLES

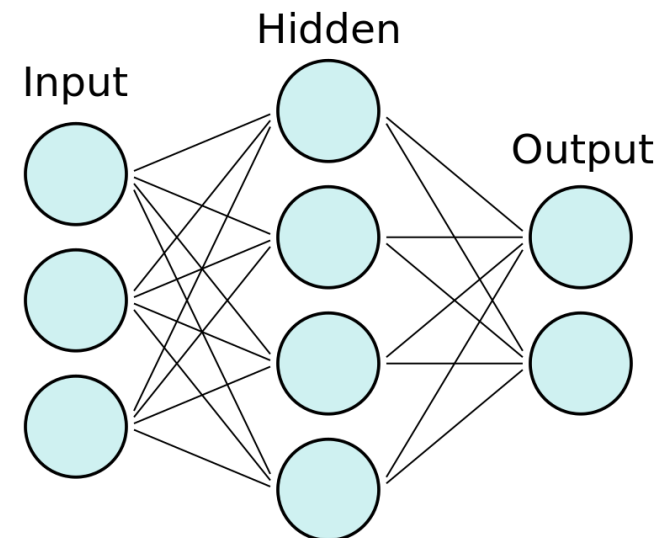
Modèle : Réseau de neurones convolutionnels

2012 = Geoffrey Hinton et son équipe lors de la compétition ImageNet Challenge

CNN présents dans de nombreux domaines : vision par ordinateur, reconnaissance vocale, NLP

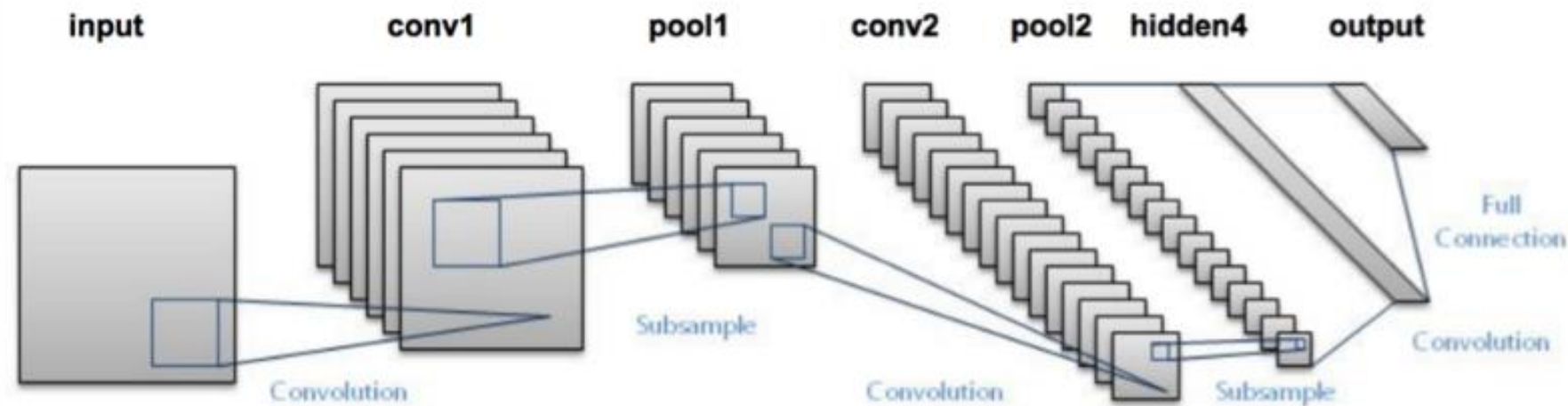
Particularité = Extraire directement les features (filtrage par convolution)

Fonction d'activation sigmoïde pour les problèmes de classification binaire



1. EVOLUTION DE LA CLASSIFICATION DES DONNEES TEXTUELLES

Principe des réseaux de neurones convolutionnels

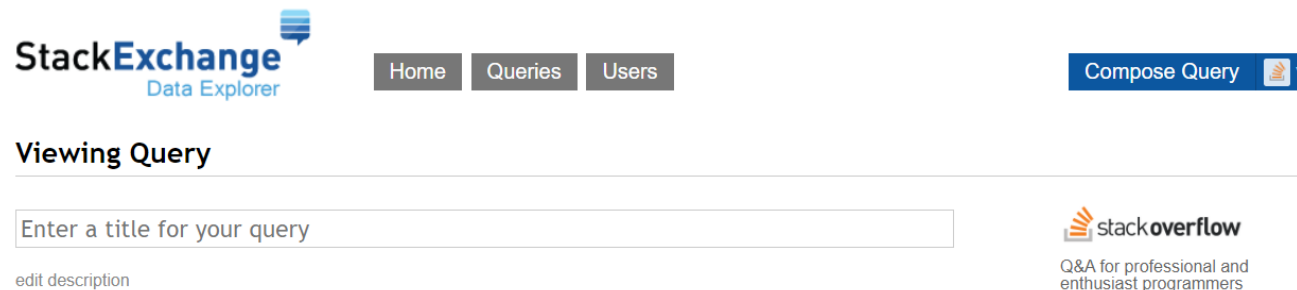


Différentes couches hiérarchisées avec chacune un rôle : Convolution, Pooling, Fully-connected (fonction d'activation sigmoïde pour classification binaire)

Couche d'Embeddings à l'entrée du réseau

2. CHOIX DES DONNEES

Données obtenues sur l'outil d'export [stackexchange explorer](#)



Base de données de 44 500 lignes et de 3 colonnes : '*Body*', '*Title*', '*Tags*'

Messages nettoyés :

- ☐ Suppression de la ponctuation et des majuscules
- ☐ Suppression des Stopwords
- ☐ Suppression des nombres seuls, des mots à 1 lettre et des espaces

2. CHOIX DES DONNEES

Algorithme de suggestion de Tags à partir de messages

	Body	Title	Tags
0	statement extract sections docx autonumbering ...	docx auto pythondocx	python
1	xarray dataset regression variables dimension ...	xarray operations dataset	python
2	number complexity optimizations check bigo pro...	prime bigo	c++
3	meaning statement section interpretations form...	meaning dot lmy	r
4	recognizer opencvcontrib line modules thing la...	attributeerror attribute createlbphfacerecognizer	python, python-3.x
5	trouble ngbootstrap dropdown component depende...	peer dependency popperjs	jquery, npm
6	vuetifys vstepper vue router requirements step...	vuetify vstepper vue router	javascript, vue.js
7	engine operations formulas operands operators ...	php parser xx	php

Données textuelles pour prédire ‘Tags’ (100 classes) à partir de ‘Body’ et ‘Title’ :
problème de classification multi-classes transformé en problème de classification multi-labels

3. METHODE BASELINE

Objectif :

Classification multi-labels pour prédire les Tags de Stack Overflow existants à partir des messages nettoyés

Tags	
0	python
1	python
2	c++
3	r



	.net-core	algorithm	amazon-web-services
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0

4 rows × 99 columns

Variable explicatives = Vectorisation des messages avec un TF-IDF

Variables à expliquer = Dummies des 100 Tags les + fréquents de Stack Overflow

Séparation des données en jeu de données d'entraînement et de données test (30%)

'estimator__alpha' : [10, 1, 0.01, 0.001, 1e-04, 1e-05, 1e-06],
'estimator__penalty' : ['l1', 'l2']

Calcul de Accuracy Score et Temps de calcul sur le jeu de données test

Entraînement du modèle OneVsRestClassifier avec SGDClassifier sur le jeu de données d'entraînement (recherche des hyperparamètres optimaux par validation croisée)

4. NOUVELLE METHODE

Encodage des données textuelles :

fit_on_texts

Création d'un dictionnaire de tous les mots entraînés sur l'ensemble des messages

```
{ '<UNK>': 1,
  'image': 2,
  'python': 3,
  'array': 4,
  'case': 5,
  'api': 6,
  'thanks': 7,
  'page': 8,
  'line': 9,
```

texts_to_sequences

Encodage des messages en utilisant le mappage effectué sur l'ensemble des messages

```
statement extract sections docx
autonumbering pythondocx text docx autonumbering
[191, 2759, 1115, 1, 1, 1, 26, 1, 1]
```

pad_sequences

Même longueur pour tous les documents

```
array([[ 191, 2759, 1115, ..., 0, 0, 0],
       [   1,  368, 1125, ..., 0, 0, 0],
       [  11,  593, 1324, ..., 0, 0, 0],
       ...,
       [1489,    8,   48, ..., 0, 0, 0],
       [ 646,  562,    1, ..., 0, 0, 0],
       [   6,  458,  522, ..., 0, 0, 0]],
```

4. NOUVELLE METHODE

Réseau de neurones convolutionnels pour un problème de classification multi-labels

```
array([[0, 0, 0, ..., 0, 0, 0],  
       [0, 0, 0, ..., 0, 0, 0],  
       [0, 0, 0, ..., 0, 0, 0],  
       ...,  
       [0, 0, 0, ..., 0, 0, 0],  
       [1, 0, 0, ..., 0, 0, 0],  
       [0, 0, 0, ..., 0, 0, 0]])
```

Variables explicatives = Matrice des
séquences encodées et de taille
limitée avec `text_to_sequences` et
`pad_sequences`
Variables à expliquer =
Matrice des Tags (Encodage de la
liste des labels avec
`MultiLabelBinarizer`)

Séparation des données
en jeu de données
d'entraînement et de
données test (30%)

'embedding_dim' : [50, 100], 'rate' : [0.0, 0.5]

Calcul de l'Accuracy,
et du Temps de
calcul le jeu de
données test

Entrainement du modèle
avec `KerasClassifier` sur le
jeu de données
d'entraînement (recherche
des hyperparamètres
optimaux par validation
croisée)

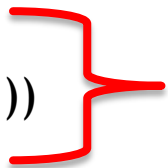
Définition de
l'architecture du CNN
à optimiser

4. NOUVELLE METHODE

Architecture du réseau de neurones convolutionnels :

```
model = Sequential()
```

```
model.add(Embedding(vocab_size, embedding_dim, input_length=maxlen))
```



1 couche d'Embeddings

```
model.add(Conv1D(num_filters, kernel_size, activation='relu'))
```

```
model.add(GlobalMaxPooling1D())
```

```
model.add(Dropout(rate))
```



1 couche de Convolution et ReLu et 1 couche de Pooling

```
model.add(Flatten())
```

```
model.add(Dense(units, activation='relu'))
```

```
model.add(Dense(100, activation='sigmoid'))
```



1 couche Fully-connected

```
model.compile(optimizer=keras.optimizers.Adam(lr), loss='binary_crossentropy', metrics=['accuracy'])
```

4. NOUVELLE METHODE

Résultats de la validation croisée pour la recherche des hyperparamètres optimaux :

Sur '*Body*'

'embedding_dim' : 100, 'rate' : 0.5

Layer (type)	Output Shape	Param #
embedding_17 (Embedding)	(None, 579, 100)	6973600
conv1d_17 (Conv1D)	(None, 575, 128)	64128
global_max_pooling1d_17 (Glo	(None, 128)	0
dropout_17 (Dropout)	(None, 128)	0
flatten_17 (Flatten)	(None, 128)	0
dense_34 (Dense)	(None, 32)	4128
dense_35 (Dense)	(None, 100)	3300
Total params: 7,045,156		
Trainable params: 7,045,156		
Non-trainable params: 0		

Sur '*Title*'

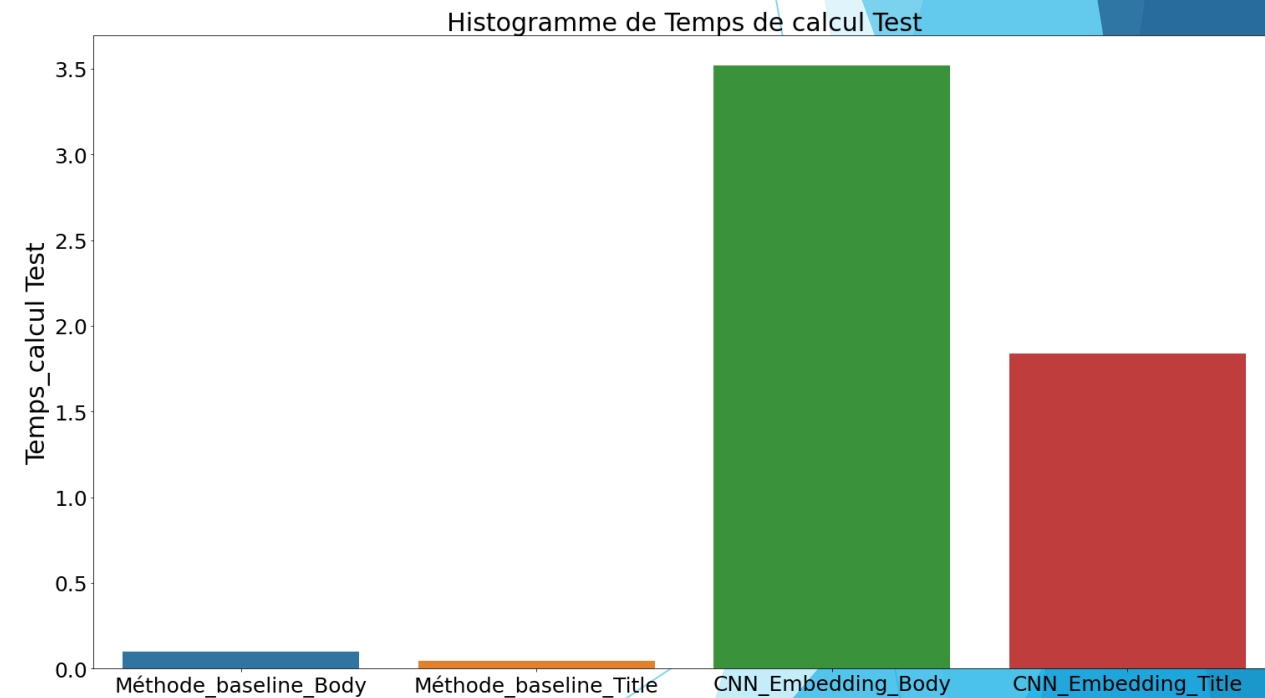
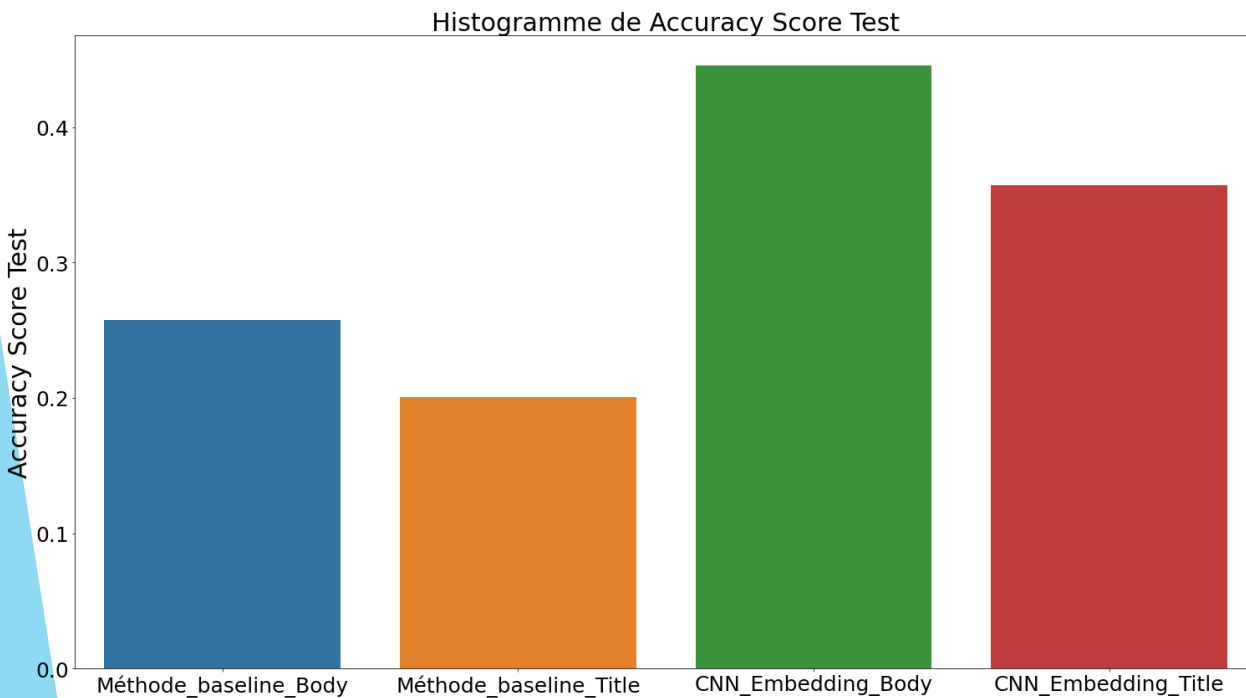
'embedding_dim' : 50, 'rate' : 0.5

Layer (type)	Output Shape	Param #
embedding_35 (Embedding)	(None, 12, 50)	921450
conv1d_35 (Conv1D)	(None, 8, 128)	32128
global_max_pooling1d_35 (Glo	(None, 128)	0
dropout_35 (Dropout)	(None, 128)	0
flatten_35 (Flatten)	(None, 128)	0
dense_70 (Dense)	(None, 32)	4128
dense_71 (Dense)	(None, 100)	3300
Total params: 961,006		
Trainable params: 961,006		
Non-trainable params: 0		

4. NOUVELLE METHODE

Comparaison des résultats avec la méthode baseline :

- Calcul du score Accuracy et du Temps de calcul sur le jeu de données test



Meilleurs résultats obtenus avec la nouvelle méthode : Accuracy = 0.445300 ('Body') et 0.356776 ('Title')

CONCLUSION

Objectif : Algorithme de suggestion de Tags à partir du message d'un utilisateur

Trouver une nouvelle approche pour améliorer les performances = **Word Embeddings**



Avantages :

- Représentation du sens des mots
- Vecteur de plus petite taille et dense
- Améliorations possibles : modèles pré-entraînés : *glove*, *BERT*, ...



Inconvénients :

- Temps plus long pour entraîner le modèle
- Technique « boîte noire »

MERCI DE VOTRE ATTENTION

QUESTIONS - REPONSES