Natural Language Processing

Détection des spam dans les emails

PRÉSENTÉ PAR MÉLODY , ZOHRA & IMEN

SOMMAIRE

- CONTEXTE
- ANALYSE DES DONNÉES
- VISUALISATIONS
- DÉFINIR L'EMBEDDING
- DÉMONSTARTION DES DIFFÉRENTS MODÈLES
- INTERFACE UTILISATEUR
- CONCLUSION

Contexte

APERÇU DE L'ENCADREMENT DU PROJET

Nous travaillons dans l'équipe data d'une agence data spécialisée en natural language processing (NLP). Notre agence vient de gagner un appel d'offres avec 6 projets différents.

Pour répondre à cet appel d'offre, le chef data officer a créé 6 équipes avec un chef de projet. Notre équipe est composée d'un data analyst, un data scientist et un data engineer.

Analyse des données

5796
Lignes

3
Colonnes

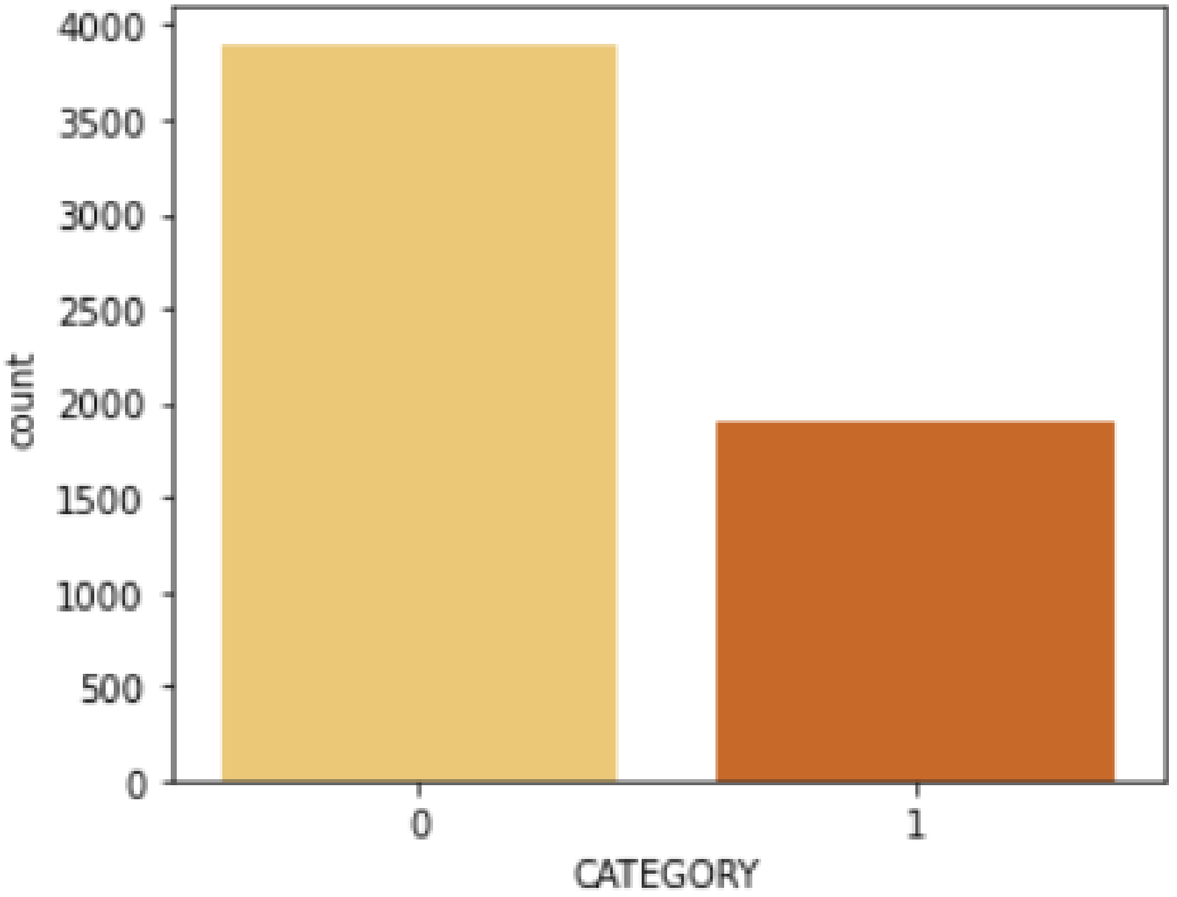
0
Valeurs
manquantes

0
Duplicates

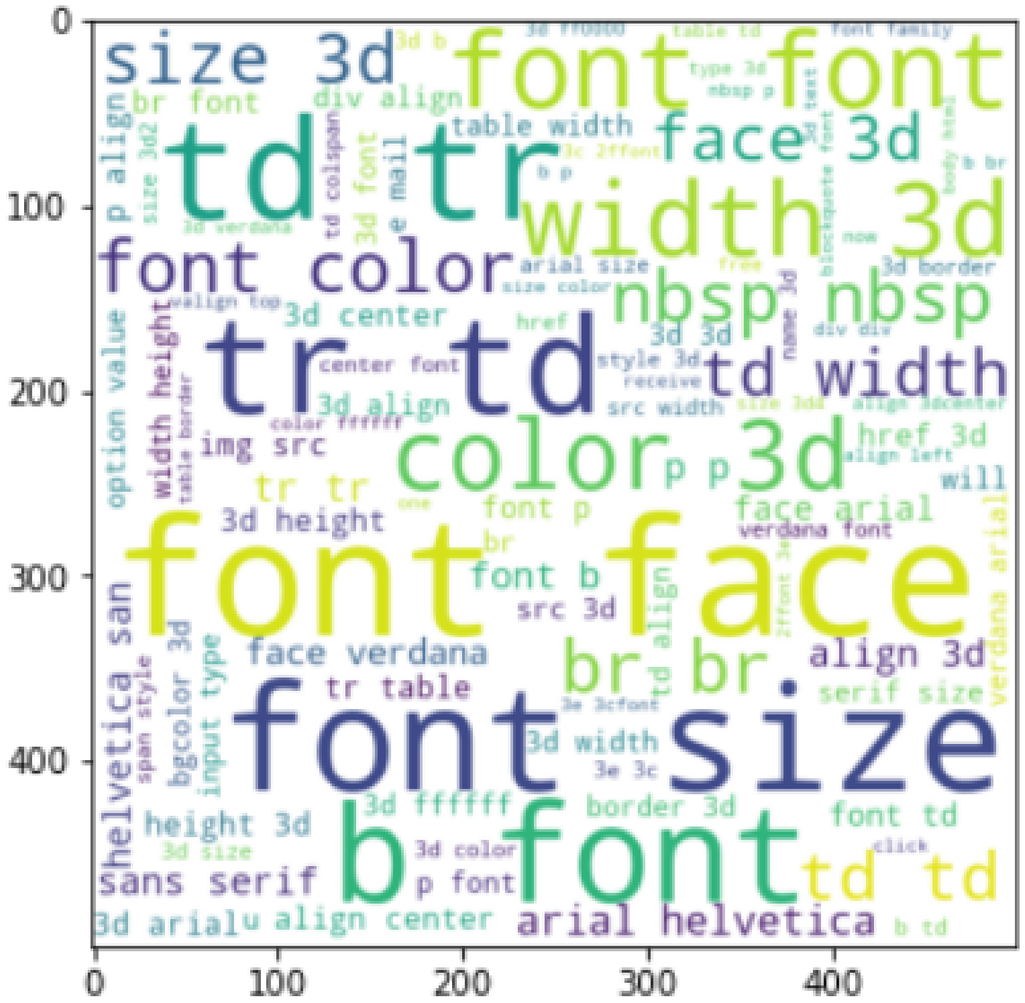
67.3% de Ham

32.7% de Spam

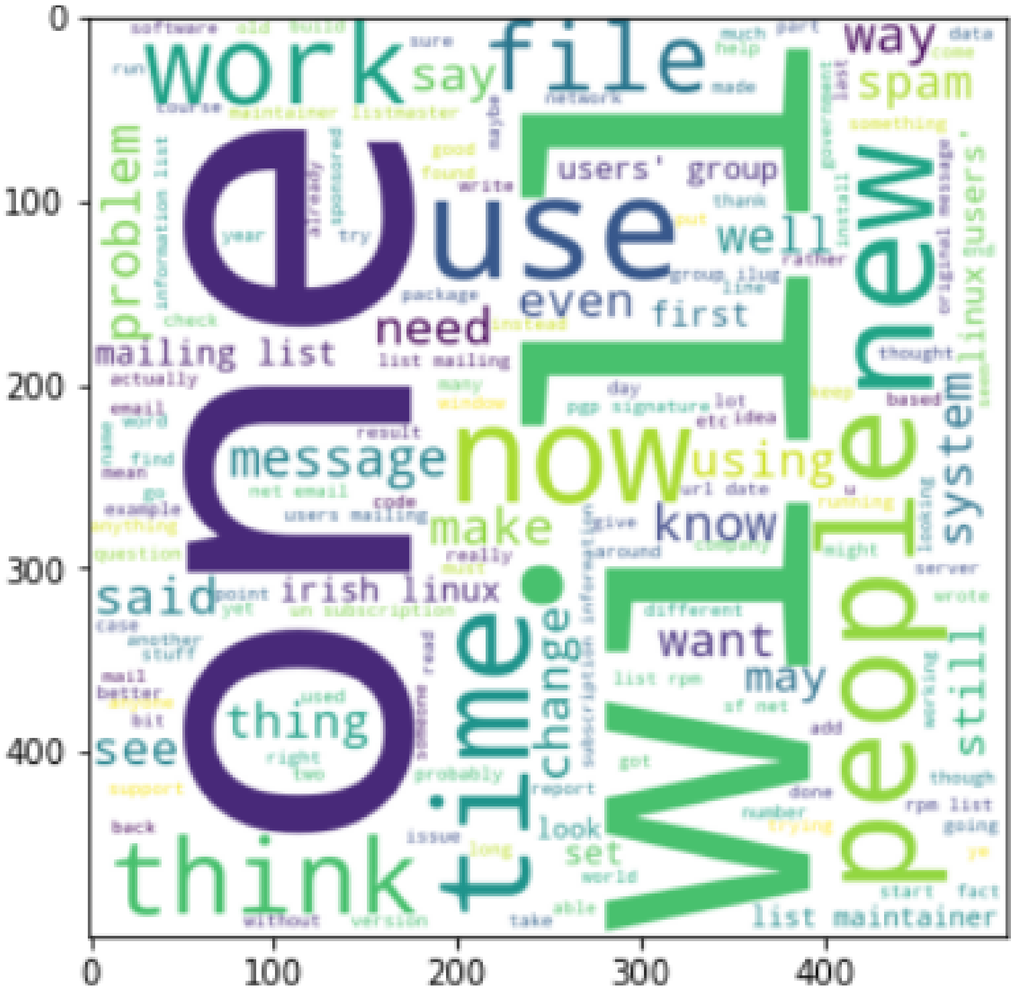
RÉPARTITION DES DONNÉES



NUAGES DE MOTS



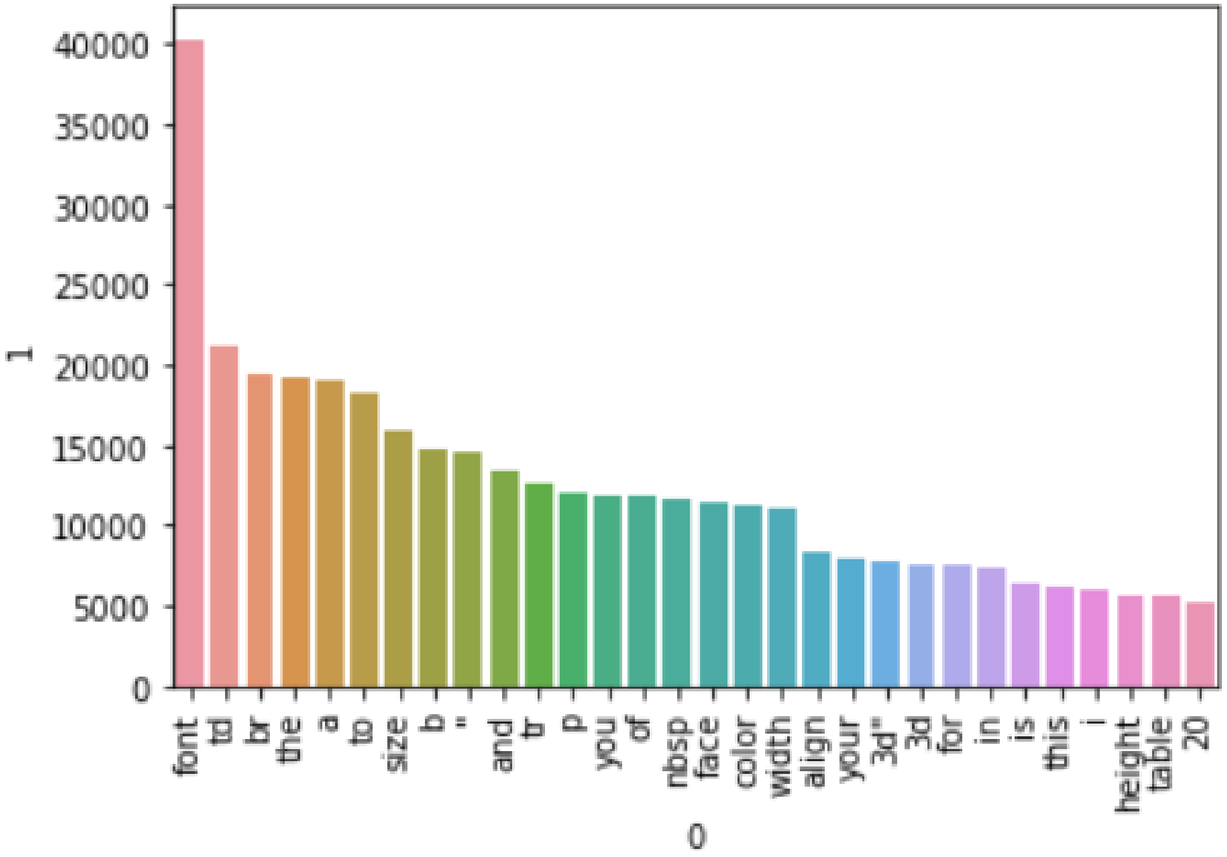
SPAM



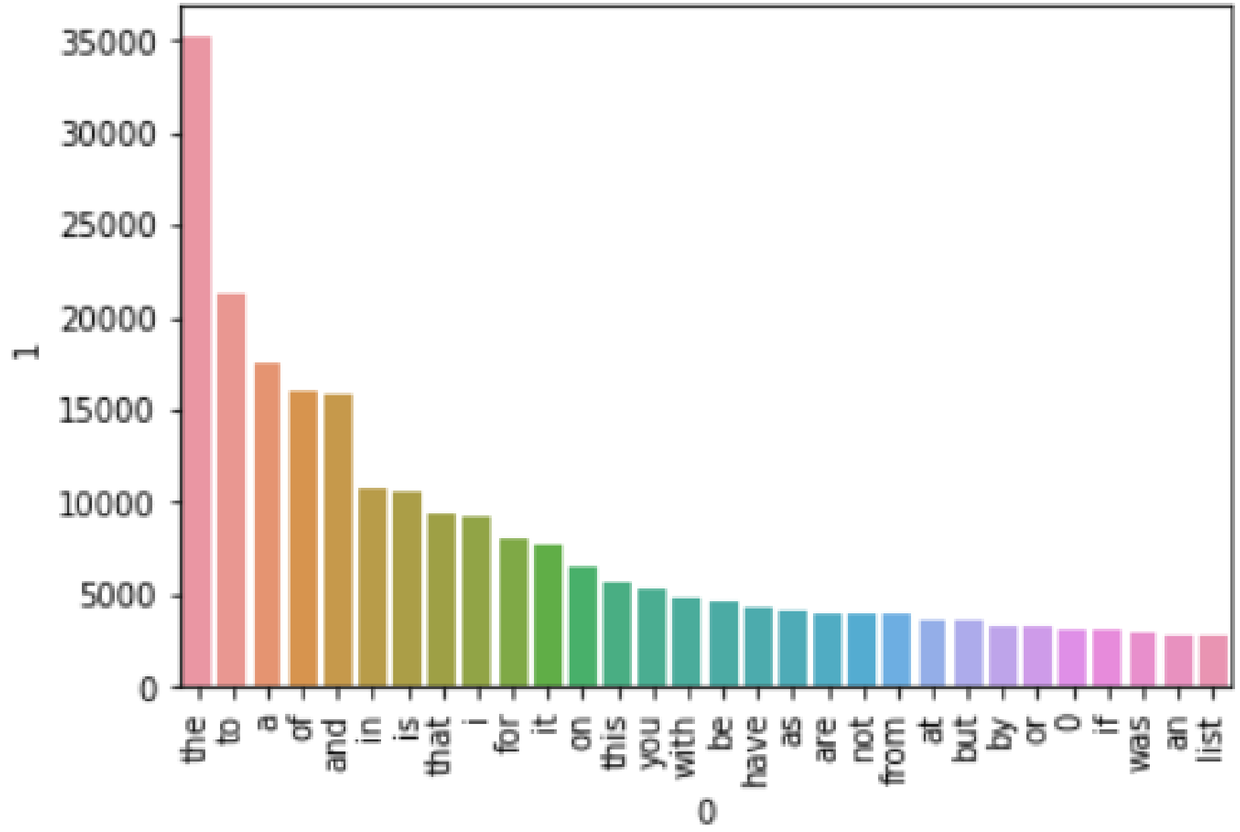
HAM

BARPLOT

Répartitions des 20 premiers mots sur chaques catégories

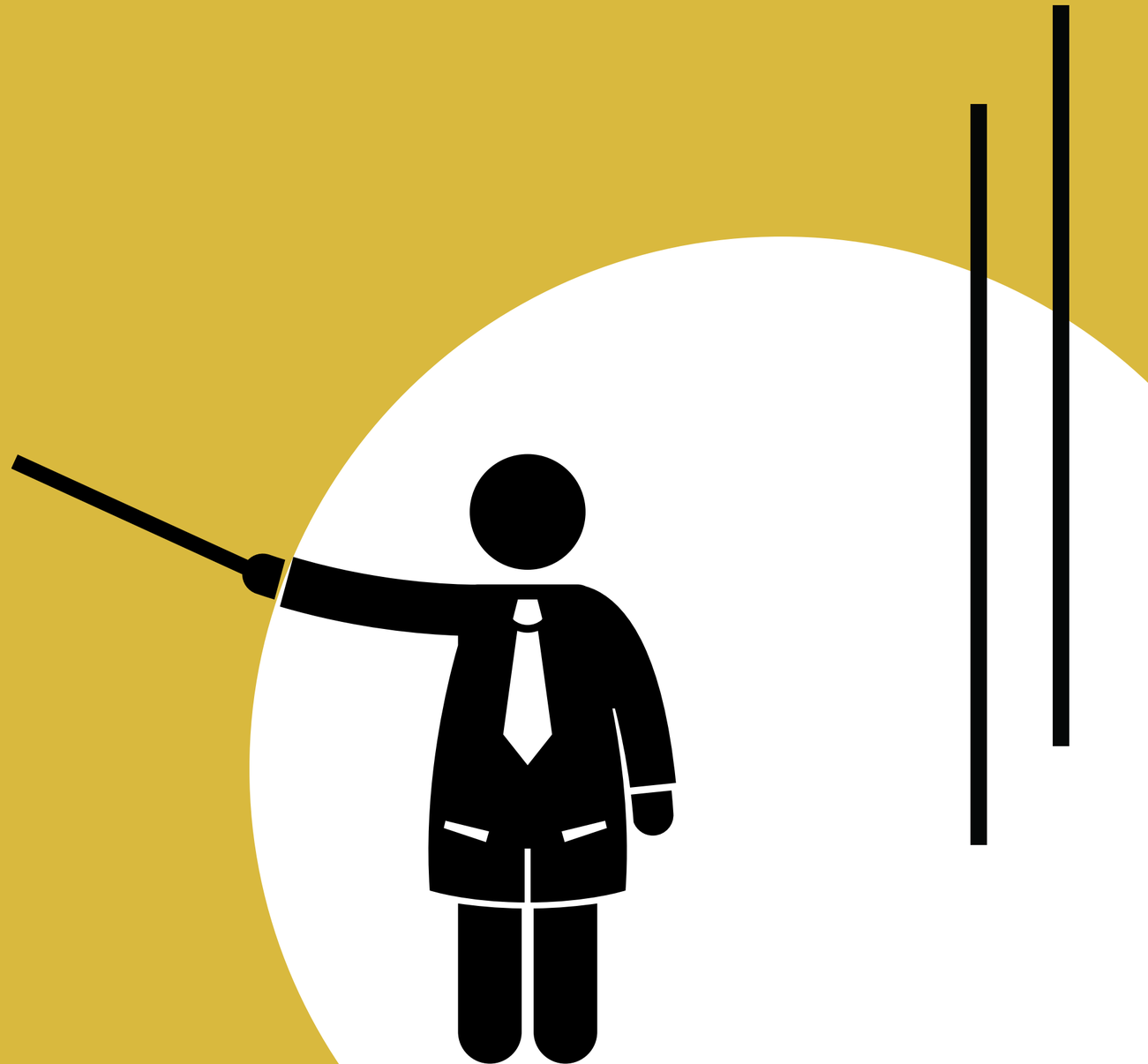


SPAM



HAM

Explication du fonctionnement de l'Embedding



Illustration


Shrek


Incredibles


The Triplets of Belleville


Harry Potter

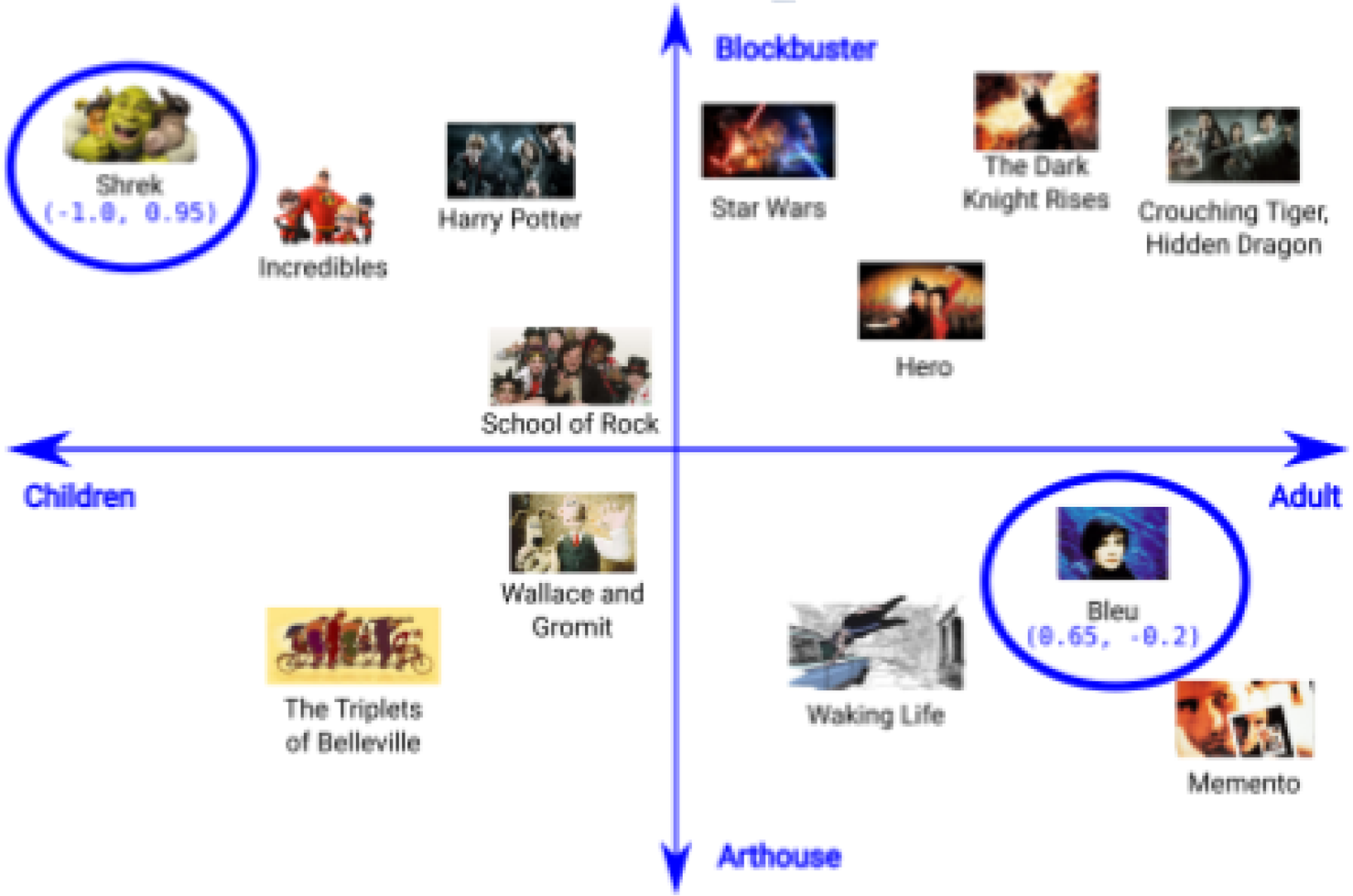

Star Wars


Bleu

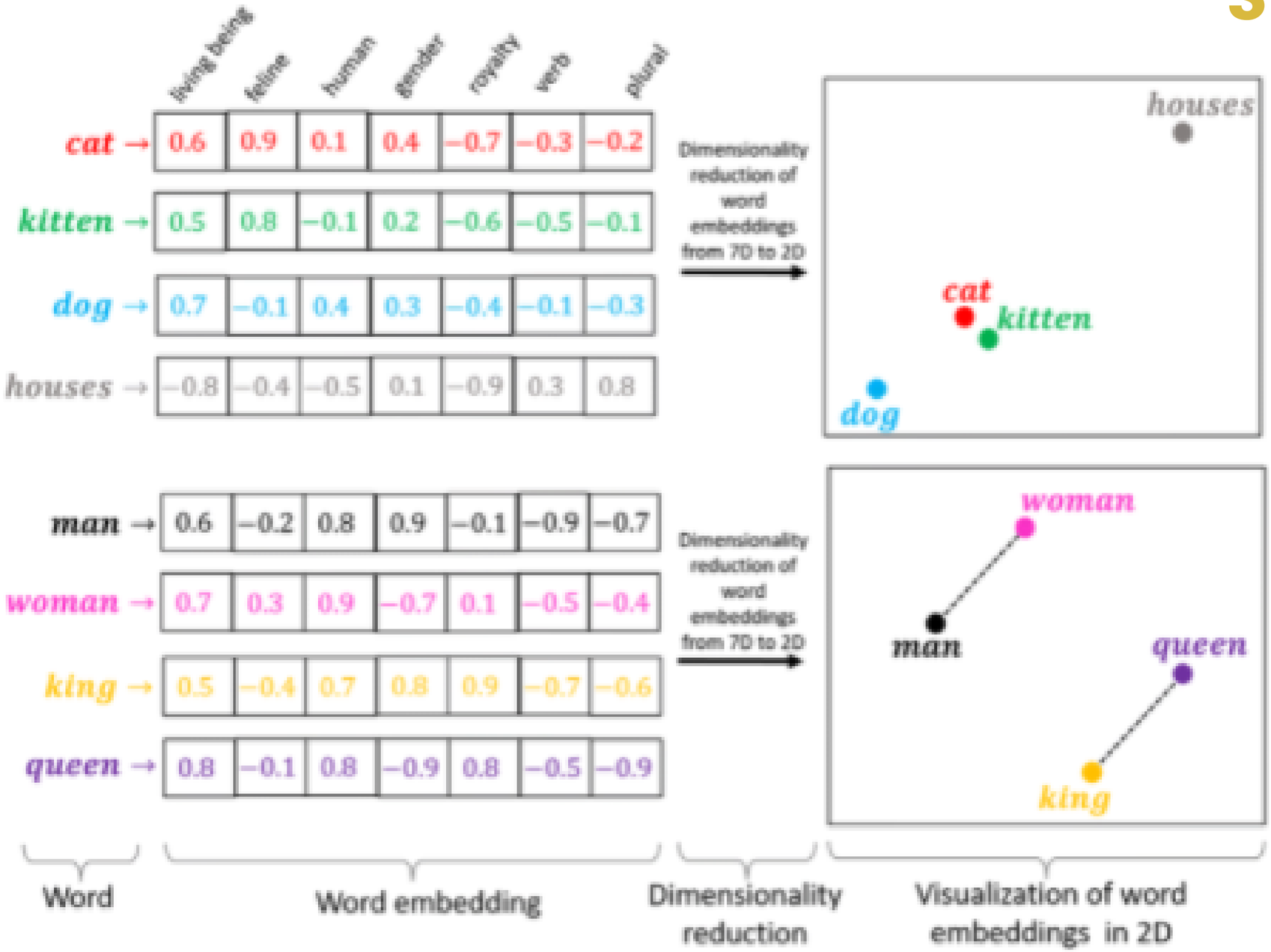

The Dark Knight Rises


Memento

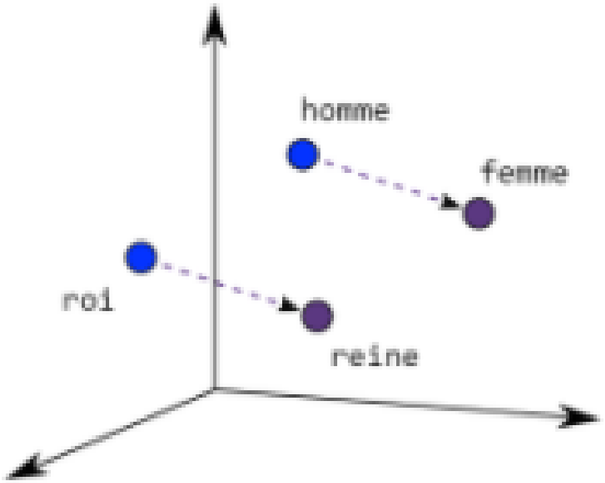
De 1 dimension à 2 dimension



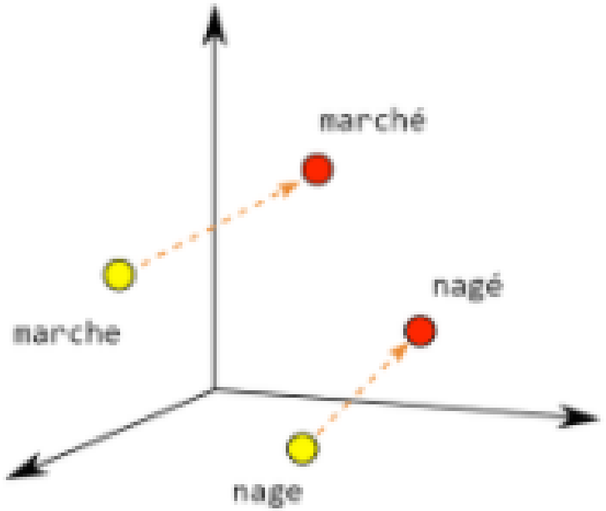
Signification des dimensions



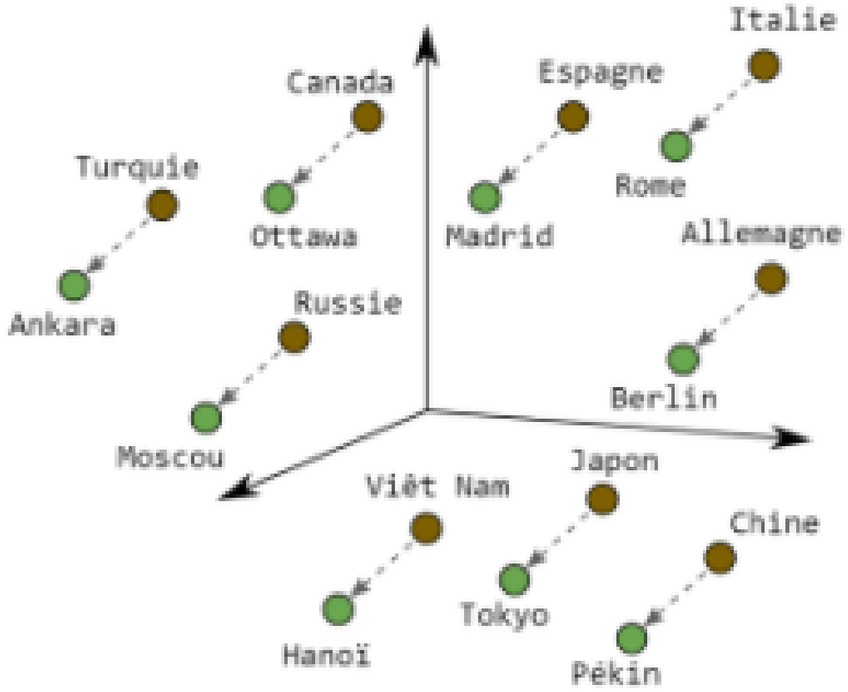
Comparaisons selon les dimensions



Homme – Femme



Temps du verbe



Pays – Capitale

Démonstrations de l'Embedding

DÉFINIR CE QU'IL SE PASSE

Exemple mots :	BOnjour	Monsieur	je	vous	...					
	Hello	dear	monsieur	je	vous	contact	pour	...		
Tokenizer :										
Monsieur	1			5	1	2	3			
Hello	4									
Bonjour	5			4	8	1	2	3	6	7
dear	8									
je	2									
vous	3									
contact	6									
pour	7			**Changement de dimension :**						
				Monsieur	0.3	0.1	0.4	0.3	0.9	
				Hello	-0.3	0.7	0.8	0.1	0.03	

Avant le changement de dimension les vecteurs ont la même shape

Exemple mots :	BOnjour	Monsieur	je	vous	…					
	Hello	dear	monsieur	je	vous	contact	pour	…		
Tokenizer :										
Monsieur	1			5	1	2	3	0	0	0
Hello	4									
Bonjour	5			4	8	1	2	3	6	7
dear	8									
je	2									
vous	3									
contact	6									
pour	7			**Changement de dimension :**						
				Monsieur	0.3	0.1	0.4	0.3	0.9	
				Hello	-0.3	0.7	0.8	0.1	0.03	

Première itération modèle avec Embedding

COUCHES MODÈLE

```python
inputs = tf.keras.Input(shape=(14804,))

embedding = tf.keras.layers.Embedding(
    input_dim=30000,
    output_dim=64
)(inputs)

flatten = tf.keras.layers.Flatten()(embedding)
outputs = tf.keras.layers.Dense(1, activation='sigmoid')(flatten)
model = tf.keras.Model(inputs=inputs, outputs=outputs)

#compile
model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=[
        'accuracy',
        tf.keras.metrics.AUC(name='auc')
    ]
)
```

MODÈLE SUMMARY

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 14804)]           0

 embedding (Embedding)       (None, 14804, 64)         1920000

 flatten (Flatten)           (None, 947456)            0

 dense (Dense)               (None, 1)                 947457

=================================================================
Total params: 2,867,457
Trainable params: 2,867,457
Non-trainable params: 0
_________________________________________________________________
None
```

```python
history = model.fit(
    X_train,
    y_train,
    validation_split=0.2,
    batch_size=32,
    epochs=100,
    callbacks=[
        tf.keras.callbacks.EarlyStopping(
            monitor='val_loss',
            patience=3,
            restore_best_weights=True
        )
```

Résultats :

Test Loss : 0.0222

Test Accuracy : 99.37%

Test AUC : 0.9989

Seconde itération modèle avec Embedding

COUCHES MODÈLE

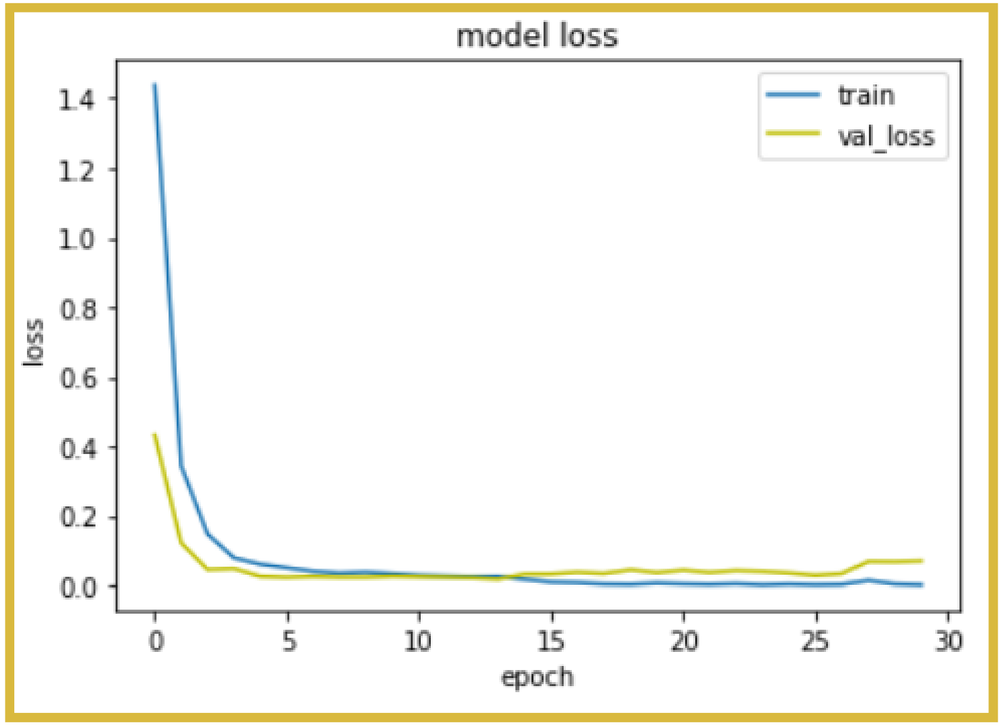
```
model = Sequential()

#Convolution
model.add(Embedding(input_dim = 30000,
                    output_dim=64,
                    input_shape=input_shape))
#Flatten
model.add(Flatten())
#Couches denses
model.add(Dense(16, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))

#Modele compile
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy', 'AUC'])

#Modele fit
history = model.fit(X_train, y_train,
                    validation_split=0.2,
                    batch_size=32,
                    epochs=30)
```

LEARNING CURVE



Résultats :

Test Loss : 0.0693

Test Accuracy : 99.10%

Test AUC : 0.9946

Modèle de Machine Learning avec LightGBMClassifier

	precision	recall	f1-score	support
0	0.99	0.99	0.99	798
1	0.99	0.97	0.98	362
accuracy			0.99	1160
macro avg	0.99	0.98	0.99	1160
weighted avg	0.99	0.99	0.99	1160

STANDARDISER

NLTK

WORDNETLEMMATIZER()

TFIDFVECTORIZER(NGRAM_RANGE=1,3)

LIGHTGBM CLASSIFIER

Démonstration de l'interface utilisateur

CONCLUSION

Axes d'améliorations

PREMIER AXE

Réussir à améliorer l'application

DEUXIÈME AXE

Déploiement sur Streamlit en réglant le problème des "versions"

TROISIÈME AXE

Meilleur nettoyage des mots pour améliorer notre corpus