



# CATEGORISEZ AUTOMATIQUEMENT DES QUESTIONS

Projet 6

Azim Makboulhousen  
23 Mai 2018



# Sommaire

- Introduction
- Les données
- Prédiction en utilisant le topic modeling
- Prédiction par apprentissage supervisé
- Résultat et implémentation
- Conclusion

# Introduction

# Objectif du projet

- Suggestion automatique de tags
- Traitement de données textuelles
- Approches supervisées et non supervisés
- Interface web



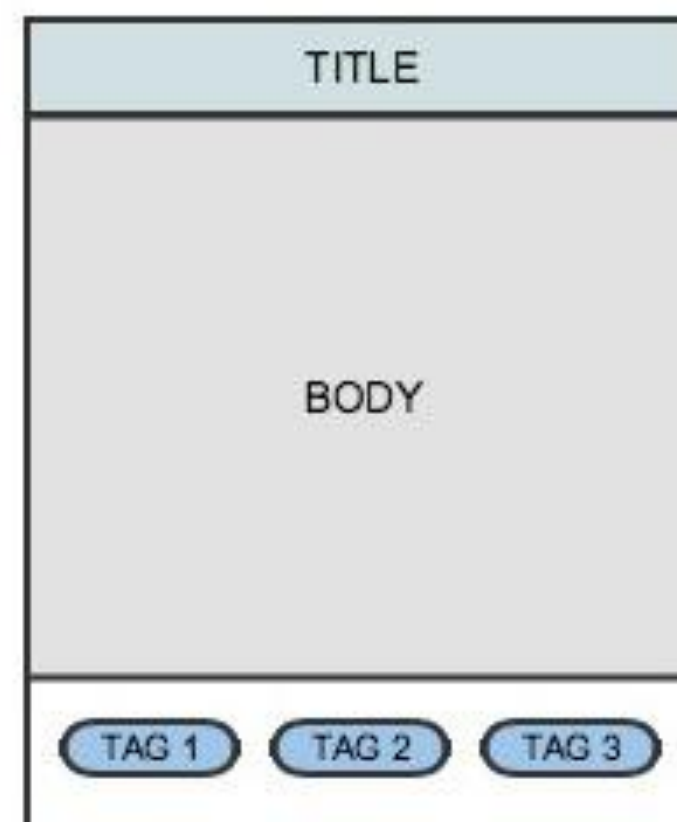
# Les données

# Collecter les données

## Récupération des questions

- Outil d'export des données Stack Overflow
- POST avec score > 5
- Limitation temps exécution → plusieurs requêtes

### POST



- Nos données = **Questions**
- Question = **Titre + Corps + Tags**
- Entre 1 à 5 tags par question
- Tag HTML dans le corps de la question



# Les données

- 64 000 questions
- Title, Body et Tags : **données textuelles**
- Pas de valeur **vide**
- Score : note attribuée au post par les utilisateurs



	TITLE	BODY	SCORE	TAGS
0	Java generics variable <T> value	<p>At the moment I am using the following code...	6	<java><generics>
1	How a value typed variable is copied when it i...	<blockquote>\n <p>Swift's string type is a va...	6	<swift><function><value-type>
2	Error while waiting for device: The emulator p...	<p>I am a freshman for the development of the ...	6	<android><android-studio><android-emulator><avd>
3	gulp-inject not working with gulp-watch	<p>I am using gulp-inject to auto add SASS imp...	10	<javascript><node.js><npm><gulp><gulp-watch>
4	React - Call function on props change	<p>My TranslationDetail component is passed an...	12	<reactjs><react-router>

# Traitement des données

- Suppression tags HTML

`<p>Hello World<br/></p>` → **Hello World**

- Unicode → ASCII

**Données enregistrées** → **Donnees enregistrees**

- Uniquement caractères alphabétiques

**Nombre > 1 occurrence** → **Nombre occurrence**

- Minuscule

**Texte Majuscule** → **texte majuscule**

- Tokenisation

**texte avec plusieurs mots** → **texte, avec, plusieurs, mots**

- Stop Words

**pour ajouter du texte** → **ajouter texte**

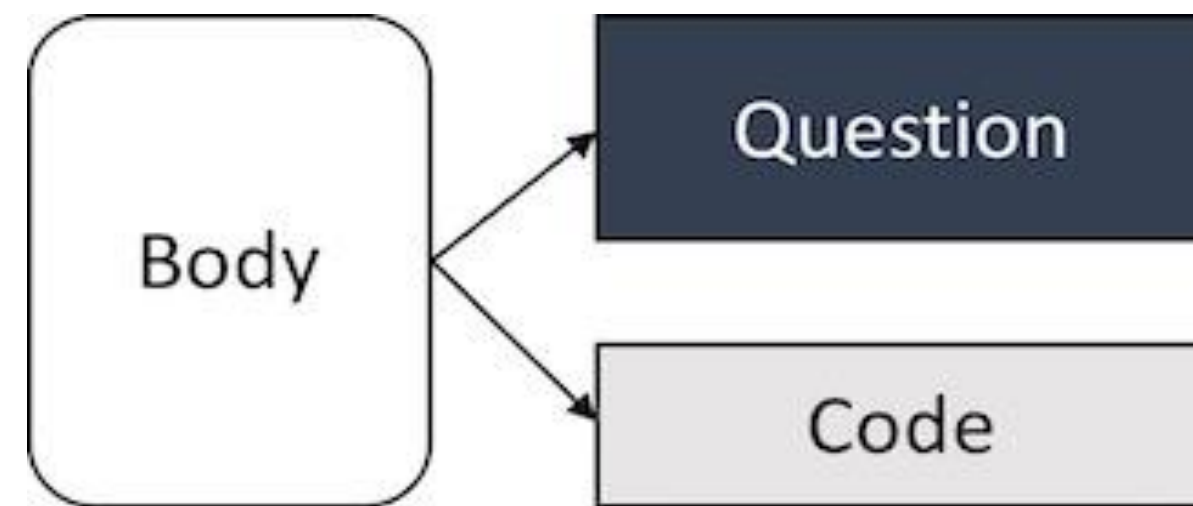
- Stemming

**joueront petites** → **jouer petit**





# Body et Tags

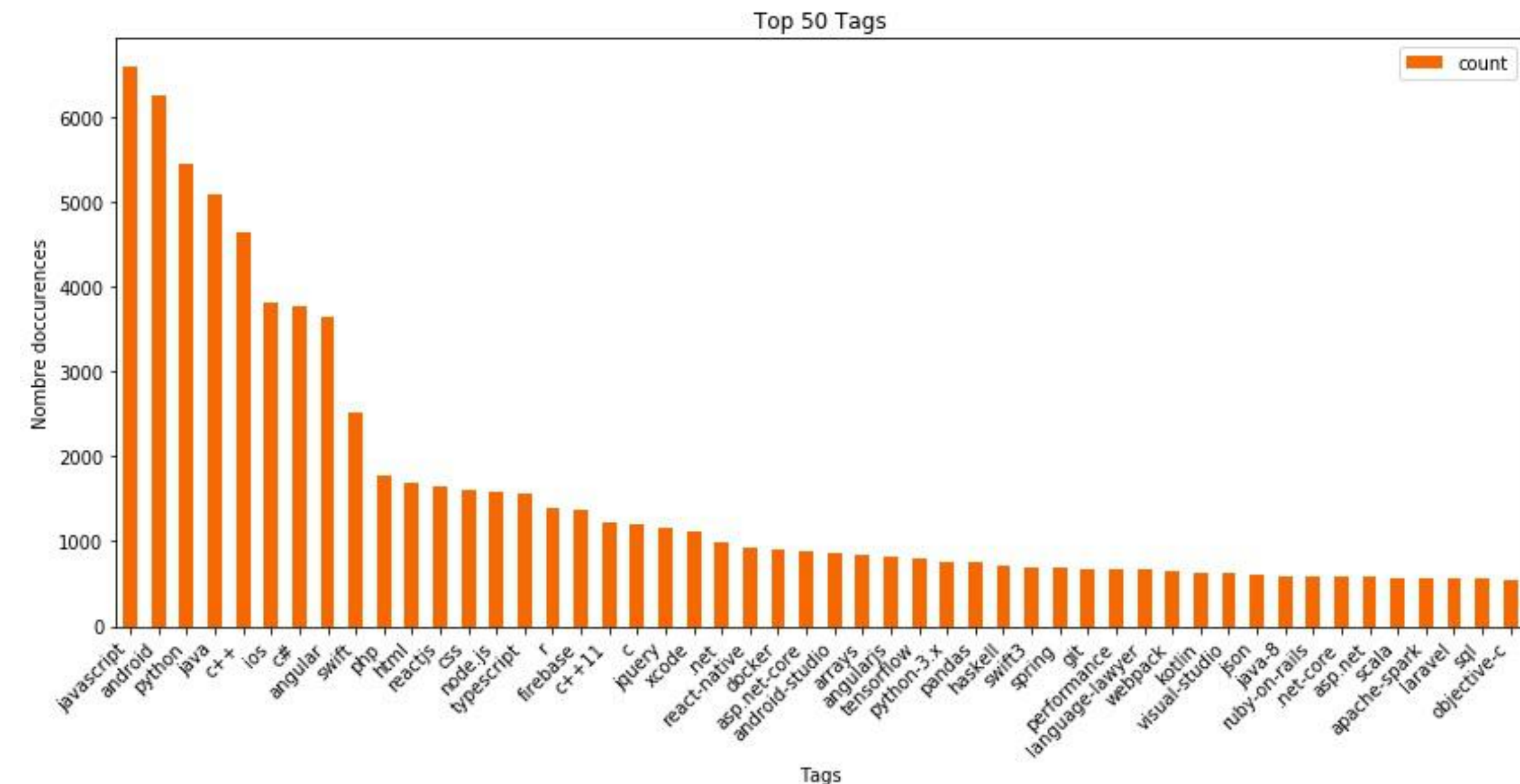


- Séparation **code** et **texte** dans BODY
- Code : suppression accents, caractères spéciaux, tokenization

## TAGS

<java><python><angular> → 

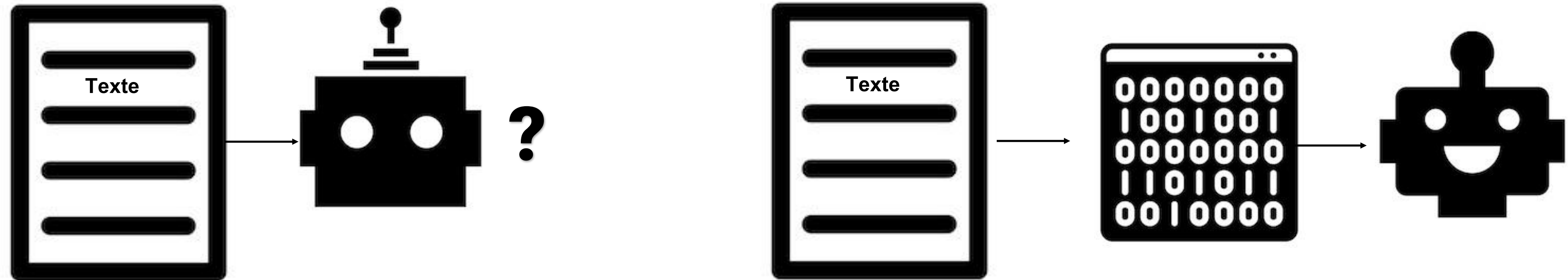
java	python	angular
------	--------	---------



javascript, android, python mots clés les plus courants

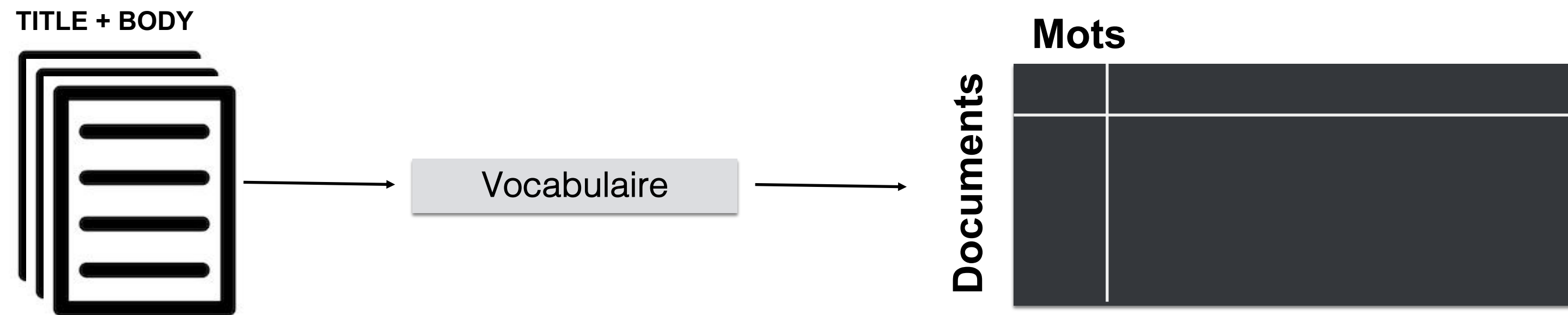
# Préparation pour modélisation

# Transformation des données



- Algorithmes d'apprentissage ne savent pas traiter du texte brut
- Il faut traduire le texte en objet interprétable
- Transformer des données textuelles en matrice
- Stratégie de modélisation sur les données transformées

# Matrices de représentation du texte



## Bag Of Words :

1. Détermination vocabulaire du corpus
2. Vecteur de document : occurrence de chaque mot
3. Matrice Documents / Mots

## N-Gramme :

- Séquence de  $n$  mots dans le corpus
- Bi-gramme : le chat, change mange, mange la souris

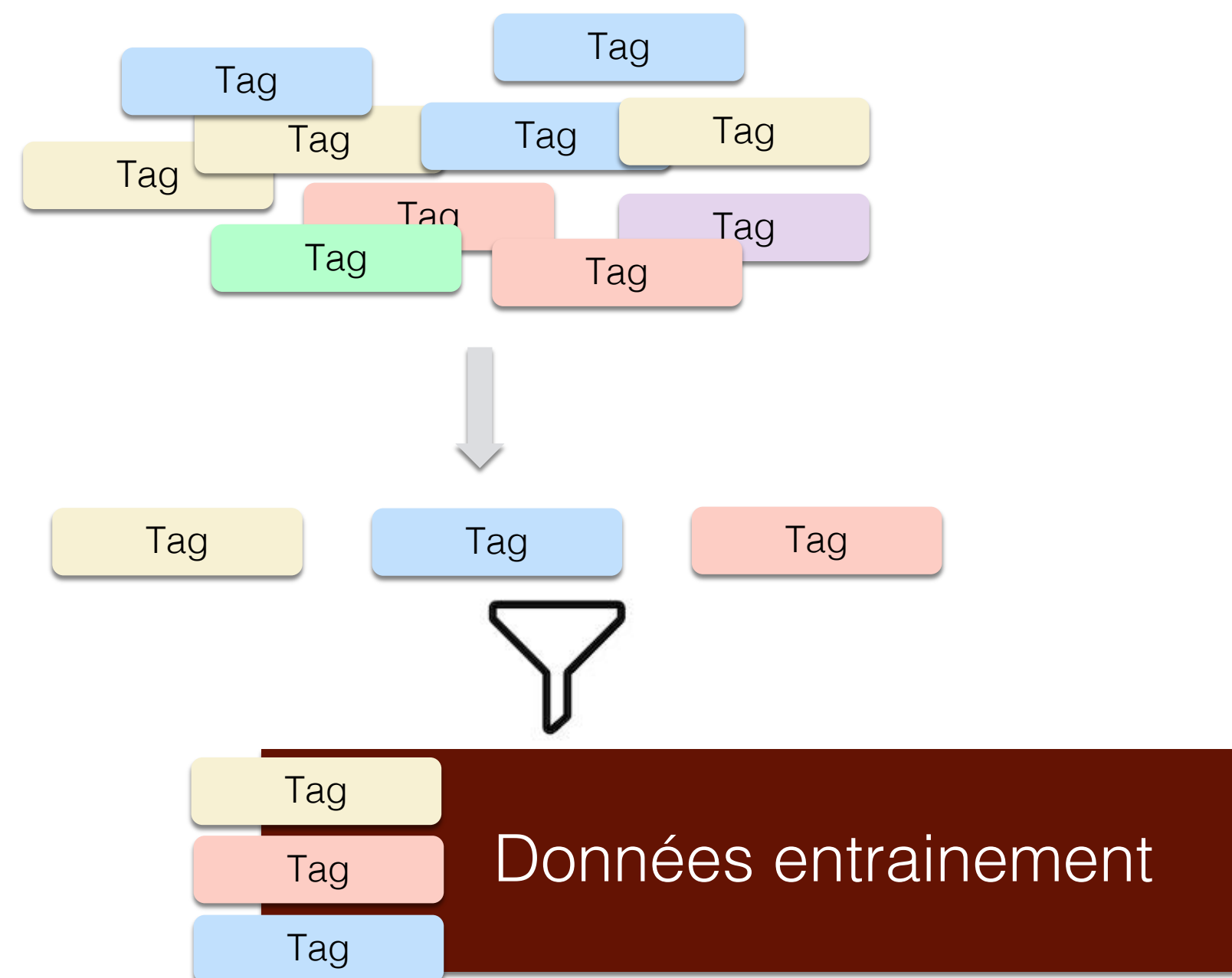
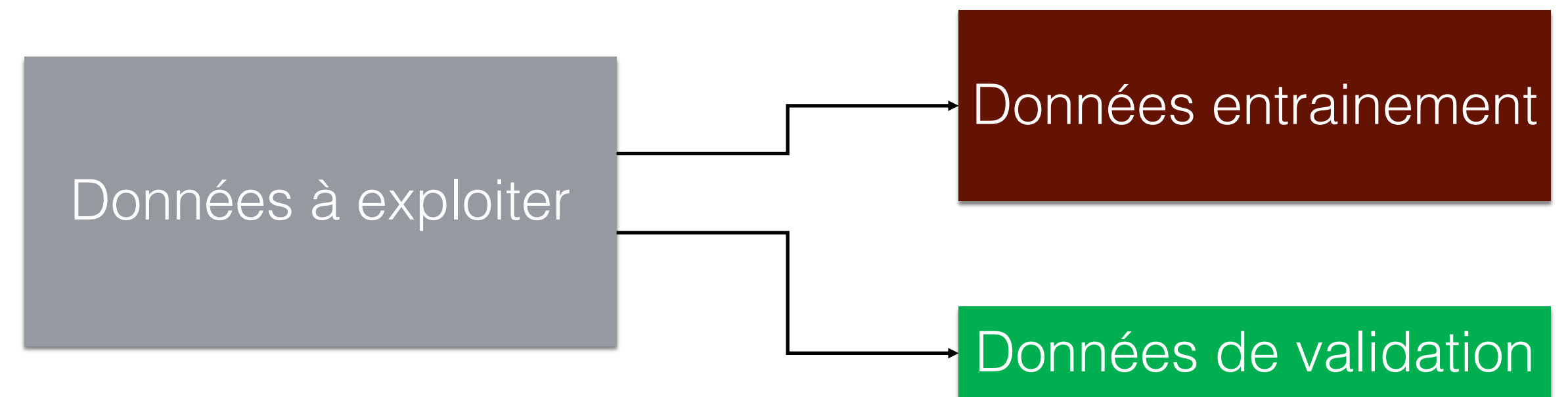
## TF-IDF :

- Méthode de pondération pour évaluer l'importance d'un document dans un document
- $TF-IDF = TF \times IDF$
- TF : Fréquence d'un mot dans le document
- IDF : Fréquence inverse du document



# Notre démarche

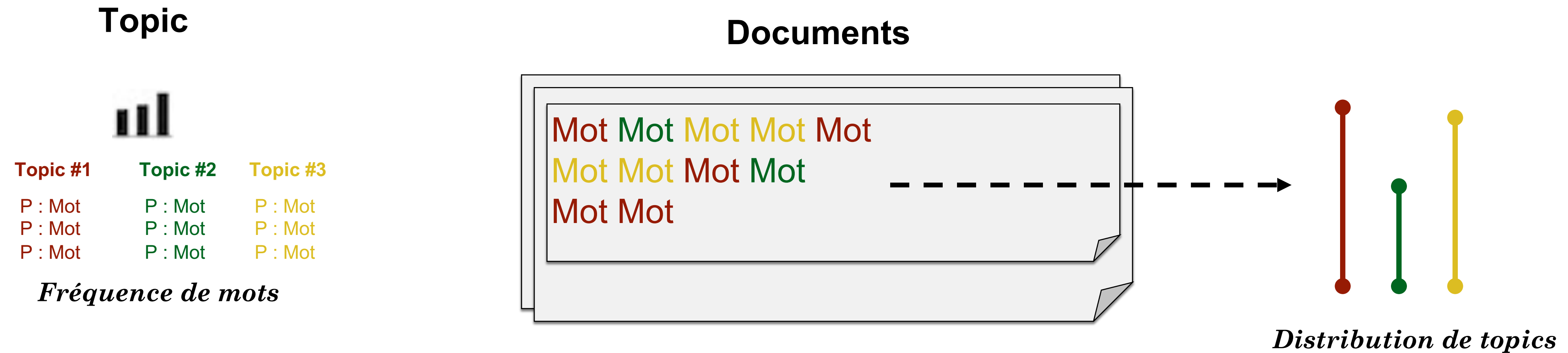
- Découpage des données en jeu d'entraînement et de validation



- Recherche des tags les plus courants
- Filtre des données sur ces tags :
  - Réduction de la volumétrie
  - Meilleure performance
  - Prédiction plus pertinente
- Données validation pas filtrées sur tags fréquents

# Apprentissage non supervisé

# Topic Modeling



- Modèle permettant de déterminer des sujets (topics) dans un ensemble de documents
- Extraction des sujets de façon non supervisée
- Capable de déterminer les sujets présents dans un document en observant tous les mots de celui-ci et en produisant une distribution des sujets

# Apprentissage

Matrice Documents/Mots

	Mot 1	Mot 2	...	Mot m
Doc 1				
Doc 2				
...				
Doc n				

Modèle

Matrice Topics/Mots

	Mot 1	Mot 2	...	Mot m
Topic 1				
Topic 2				
...				
Topic k				

Matrice Documents/Topics

	Topic 1	Topic 2	...	Topic k
Doc 1				
Doc 2				
...				
Doc n				

**Input :**

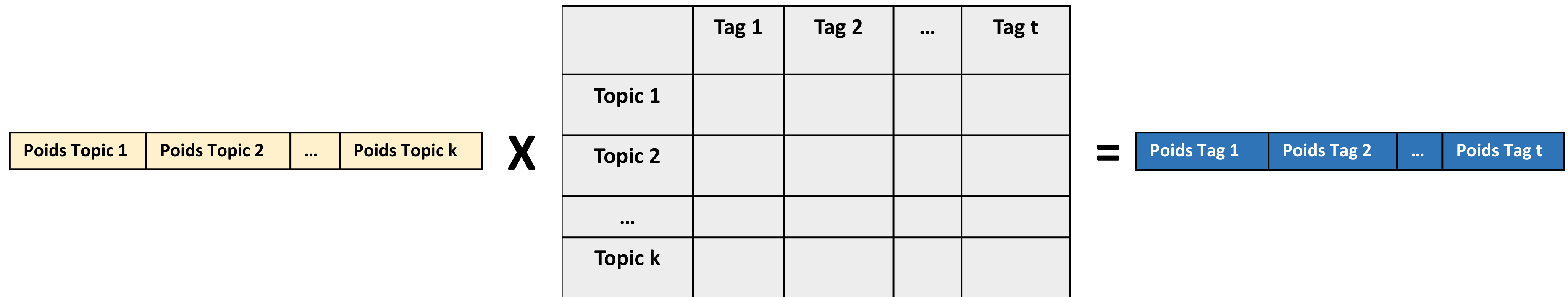
- Matrice Documents / Mots

**Output :**

- Une matrice associant les documents aux topics
- Une matrice associant les topics aux mots de notre vocabulaire



# Prédiction des tags



- Création d'une matrice Topics / Tags
  - pour chaque **tag i** :
    - chaque **topic j** :
      - SOMME probabilité d'appartenance au **topic j** des documents contenant le **tag i**
- Utilisation du modèle pour déterminer la distribution des sujets présents dans une question
- La multiplication va donner distribution des tags
- On va alors sélectionner les N tags les plus pertinents

# Les algorithmes

## Latent Dirichlet Allocation (LDA)

- modèle probabiliste
- fonctionne de manière itératif
- Bag of Words en entrée
- Recherche sur grille pour le tuning (min\_df, max\_df, nombre topics)

## Evaluation algorithme :

- Validation avec données de tests
- Calcul du score de prédiction (moyenne des score\_i)

## Non Negative Matrix Factorization (NMF)

- modèle algébrique linéaire
- factorise les vecteurs à hautes dimensions
- TF-IDF en entrée
- Tuning manuel

$$score_i = \frac{T_i}{N_i}$$

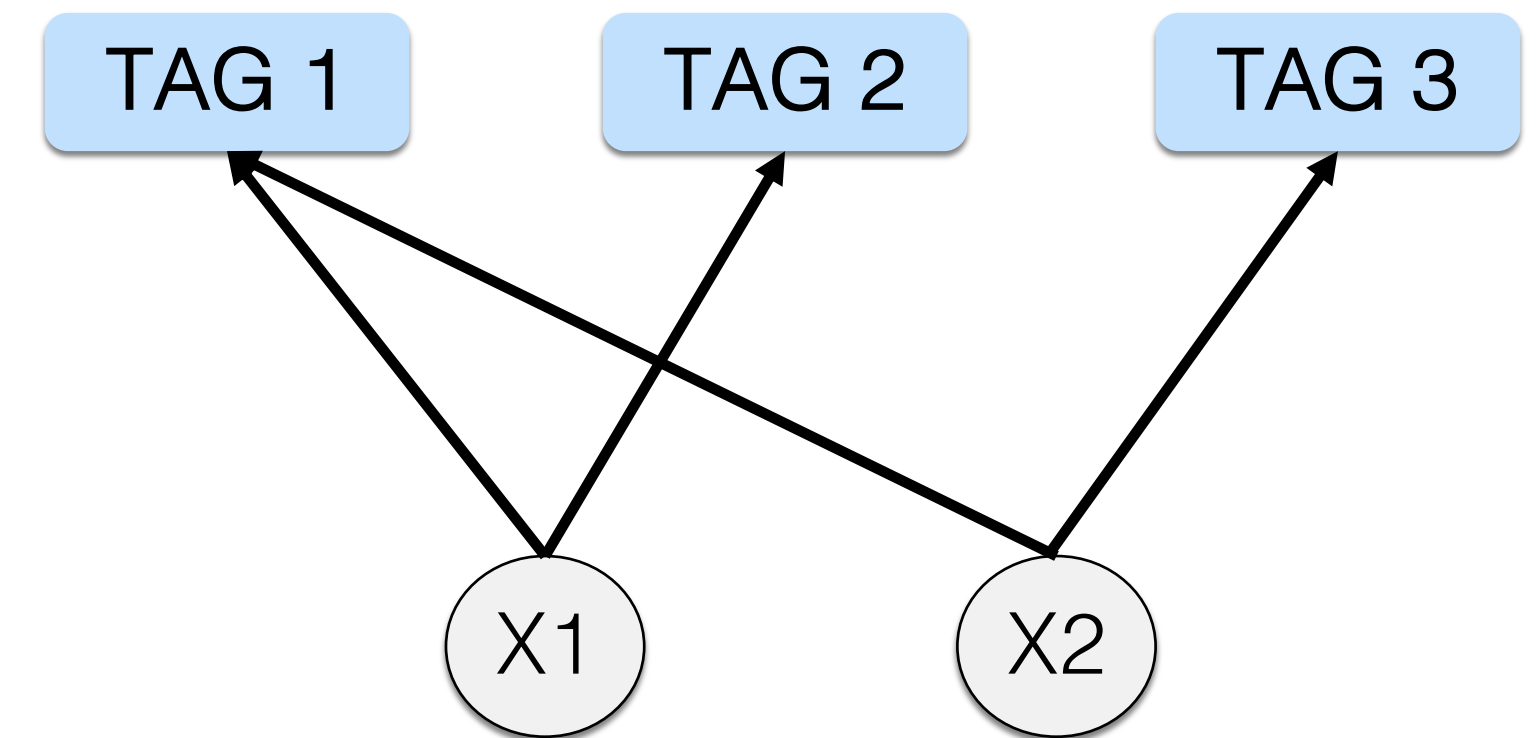
T : nombre de tags identiques aux tags réels de la question i  
N : nombre de tags total réel de la question i

# Apprentissage supervisé

# Variable cible

## Target :

- Prédiction de plusieurs tags
- Classification multi-label
- Trouver un mapping entre X et un vecteur binaire Y



## Multi label :

- Librairie **sklearn** implémente le multi-label
- Binarisation de la variable Y
- Entraînement d'un classifieur à chaque label
- Combinaison pour prédire le résultat final  
One-vs-Rest

MultilabelBinarizer

D1	Tag 1, Tag 3, Tag 0	1	1	0	1	0
D2	Tag 2, Tag 3, Tag 4	0	0	1	1	1



# Les algorithmes testés

## Input :

- Matrice Documents / Mots
  - TF-IDF
  - Unigramme et Bigramme

## Hyper-paramètres :

- min\_df,
- max\_df
- unigramme, Bigramme

**SGD (optimisation SVM)**

**Régression Logistique**

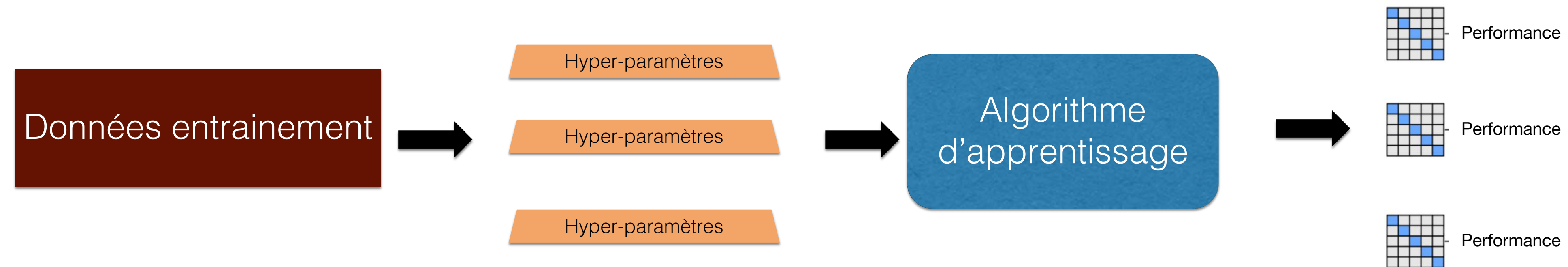
**Gaussian Naives Bayes**

**Arbre de décision**

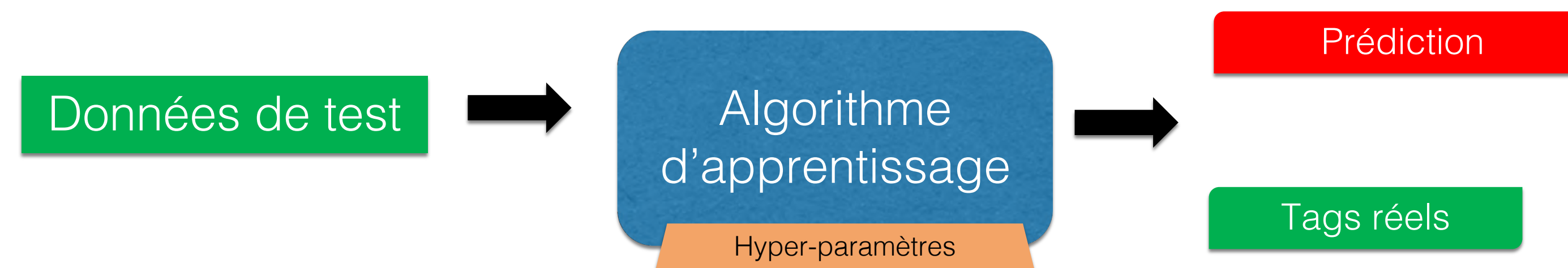
**Forêt Aléatoire**

**Gradient Boosting**

# Notre démarche d'évaluation de modèle



Evaluation de différentes valeurs d'hyper-paramètres par une recherche sur grille et une validation croisée pour trouver la meilleure performance



## Comparaison algorithmes :

- Calcul du score de prédiction (idem non supervisé)

On évalue notre algorithme sur les meilleures valeurs des hyper-paramètres avec les données de test.

# Prédiction et évaluation des algorithmes

## Prédiction :

- Prédiction  $p(\text{tag}_{k=1} | d_i)$  pour tous les  $k$  de notre liste de tags
- Sélection de  $N$  (5) tags ayant le meilleur score

## Comparaison entre les algorithmes :

- Score prédiction = équivalent à un score de rappel
- Utilisation du jeu de test

Prédiction



Réel



$$\text{score} = \frac{2}{4}$$

# Résultats et implémentation



# Résultats –Score prédiction jeu de test

## Modèles supervisés

	Gaussian Naive Bayes	Decision Tree	SGD	Random Forest	Gradient Boosting
Scores	15.64 %	50.43 %	55.01 %	40.06 %	48.07 %

## Modèles non supervisés

	LDA	NMF
Scores	24.65 %	26.83 %

- SVM Linéaire optimisé avec une descente de gradient  
➔ 55% des tags correctement prédits (5 tags)


## Tags

3	4	5	6	7	8
48,86%	52,86%	55,01%	56,21%	56,90%	57,45%

# Implémentation interface WEB

Interface disponible ici : <http://suggesttags.herokuapp.com/>

- Suggestion de 7 tags



## Ask a question - Stack Overflow

Title

sklearn and large datasets

Text

I use a lot sklearn but for much smaller datasets.

In this situations the classical approach should be something like.

Read only part of the data -> Partial train your estimator -> delete the data -> read other part of the data -> continue to train your estimator.

I have seen that some sklearn algorithm have the partial fit method that should allow us to train the estimator with various subsamples of the data.

Now I am wondering is there an easy why to do that in sklearn? I am looking for something like

Code

```
r = read_part_of_data('data.csv')
m = sk.my_model
for i in range(n):
    x = r.read_next_chunk(20 lines)
    m.partial_fit(x)

m.predict(new_x)
```

Tags

python

scikit-learn

machine-learning

r

csv

tensorflow

pandas

Suggest Tags

# Conclusion

# Conclusion

- Evaluation de différentes approches pour implémenter un système de suggestion de tags
- Sélection d'un algorithme supervisé de classification multi label
- Le topic modeling n'a pas donné les meilleurs résultats mais a permis une bonne exploration des sujets
- Axes d'amélioration :
  - Word embedding et réseaux de neurones
  - Exploitation historique des utilisateurs
  - Proposition de tags non encore utilisés

Merci à mon mentor Amine Abdaoui pour sa disponibilité, ses explications et ses précieux conseils