

CATEGORISEZ AUTOMATIQUEMENT DES QUESTIONS

Projet 6

Azim Makboulhousen
23 Mai 2018



Sommaire

- Introduction
- Les données
- Préparation modélisation
- Apprentissage non supervisé
- 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

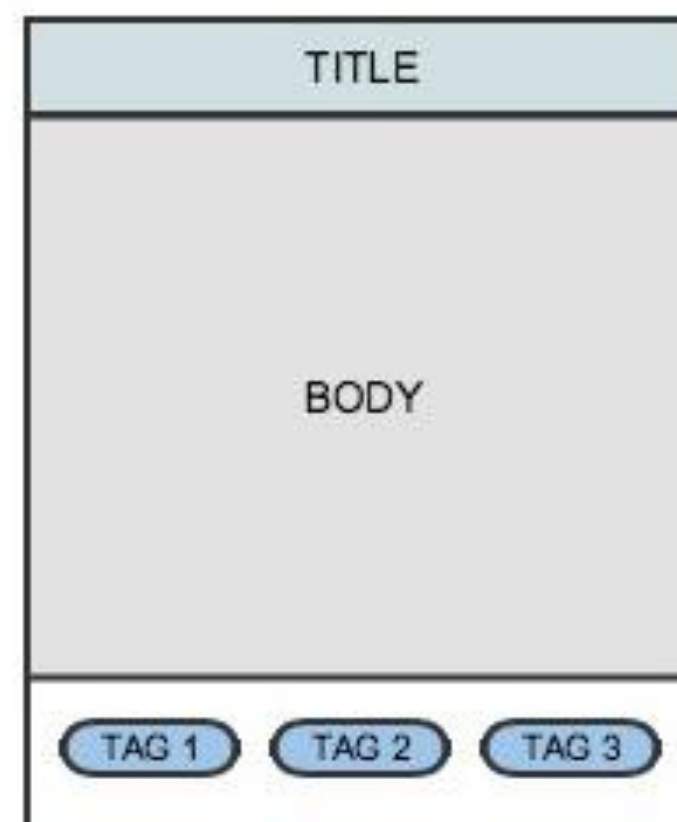
Collecter les données

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**

Nombre de caractères moyens :

- TITLE : 56
- BODY : 1780

	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
</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

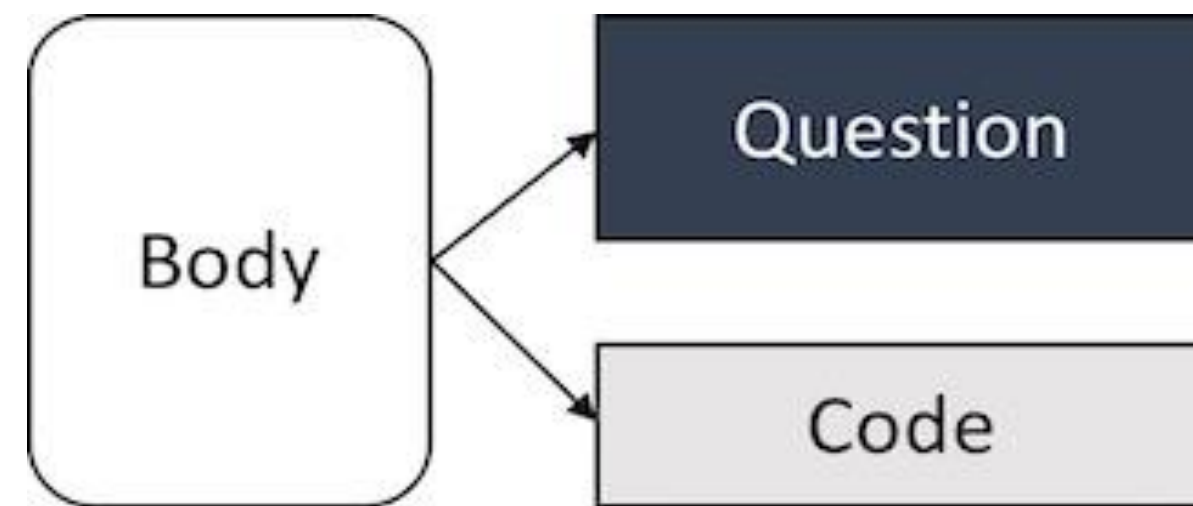
pour ajouter du texte → **ajouter texte**

- Stemming

joueront petites → **jouer petit**



Body et Tags

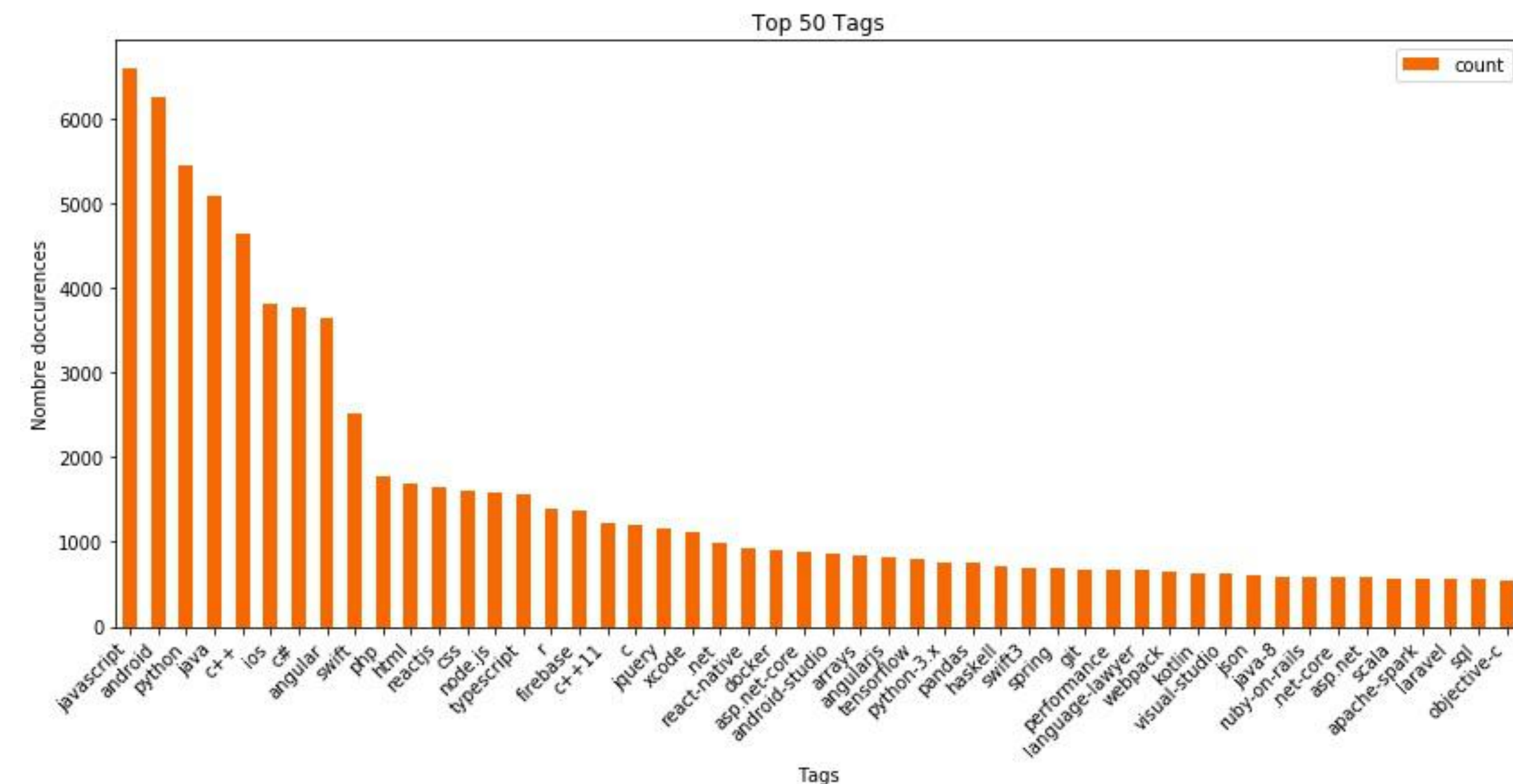


- Séparation **code** et **texte** dans BODY
- Code : suppression accents, caractères spéciaux, tokenization

TAGS

<java><python><angular> →

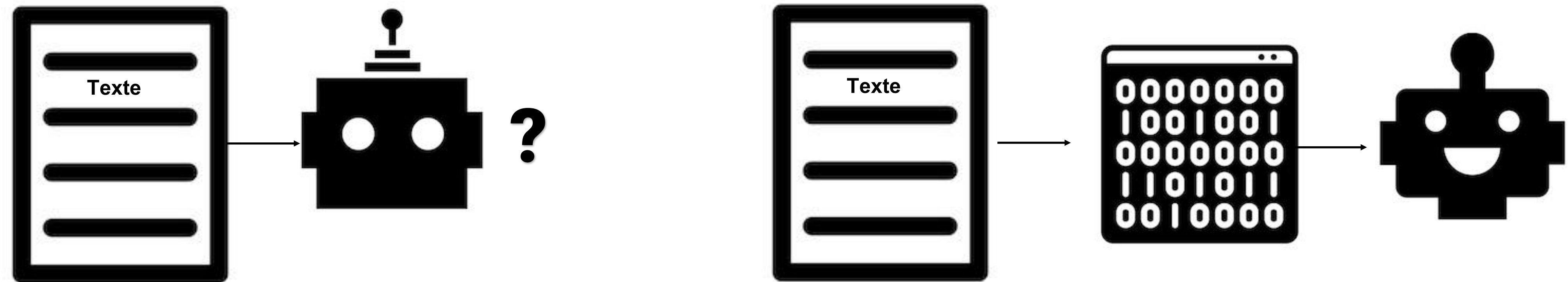
java	python	angular
------	--------	---------



javascript, android, python mots clés les plus courants

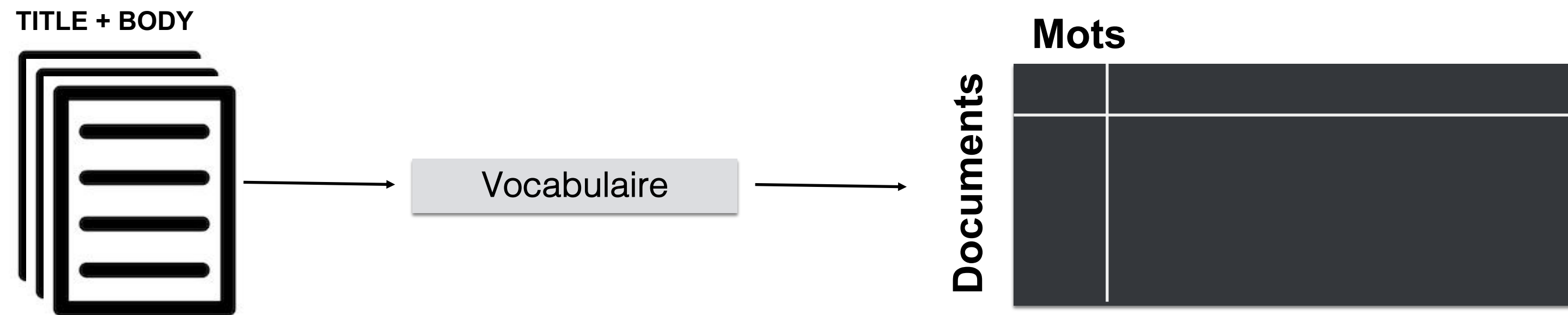
Préparation pour modélisation

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 :

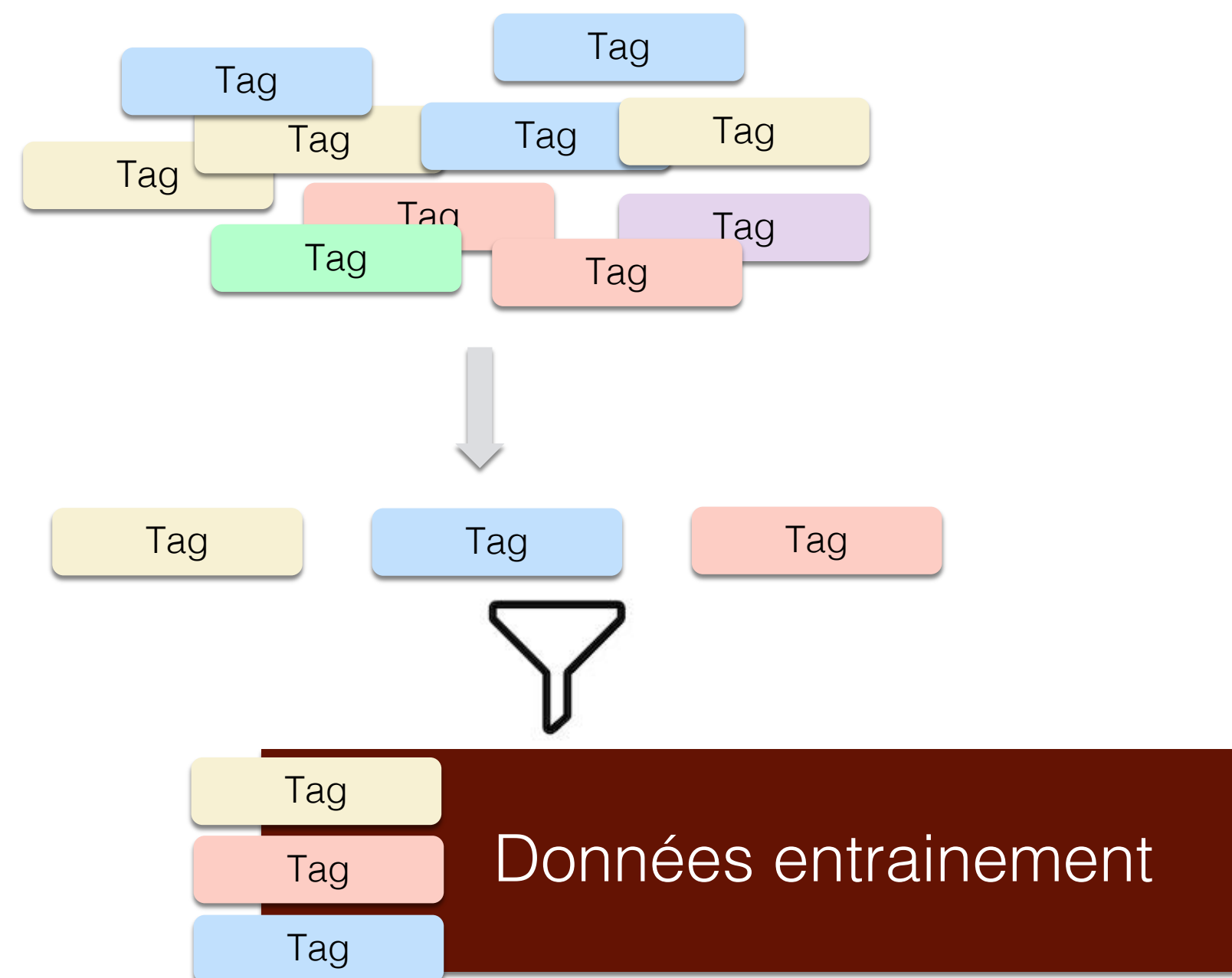
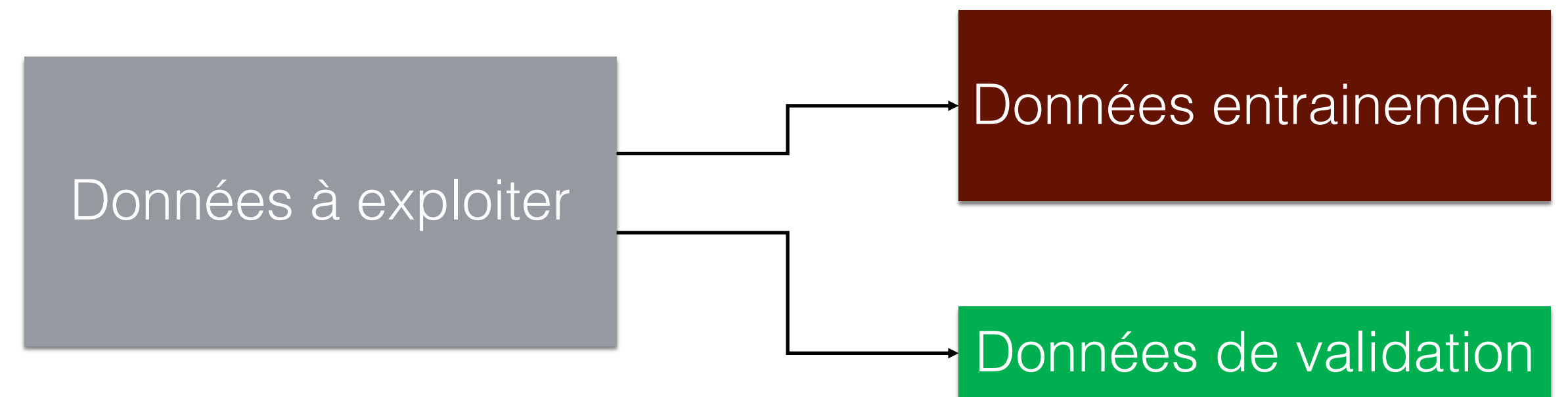
- Séquence de n mots dans le corpus
- Bi-gramme : le chat, change mange, mange la souris

TF-IDF :

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

Notre démarche

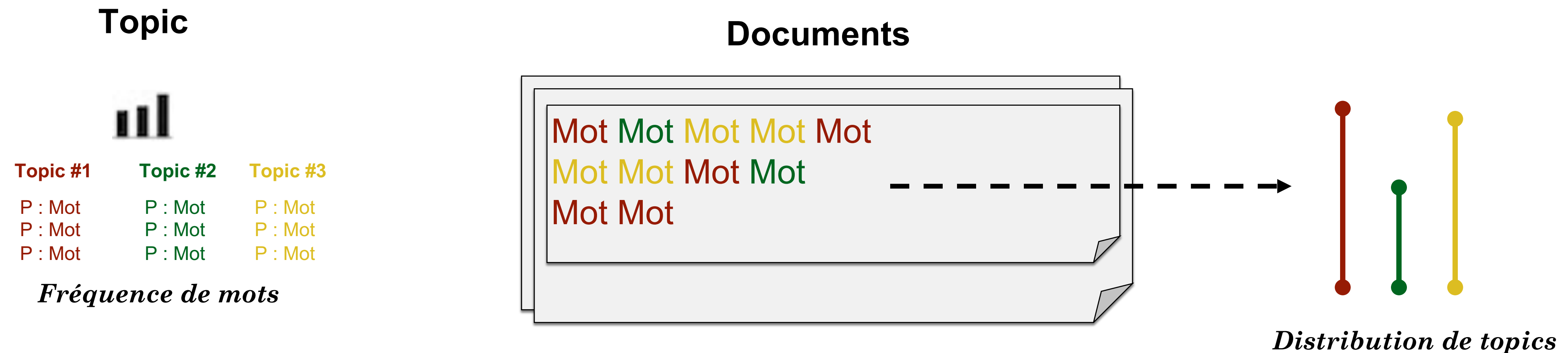
- Découpage des données en jeu d'entraînement et de validation



- 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



- Analyse statistique de texte
- Apprentissage non supervisé pour découvrir des sujets latents dans le corpus
- Permet d'assigner les sujets détectés à ces différents documents
- Donne des informations sur la sémantique d'un document.

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

Poids Topic 1	Poids Topic 2	...	Poids Topic k
---------------	---------------	-----	---------------

X

	Tag 1	Tag 2	...	Tag t
Topic 1				
Topic 2				
...				
Topic k				

=

Poids Tag 1	Poids Tag 2	...	Poids Tag t
-------------	-------------	-----	-------------

- 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

Exemple sortie LDA

	Word 0	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7	Word 8	Word 9	Word 10	Word 11
Topic 0	php	file	size	long	cs	cach	ns	clang	laravel	main	std	error
Topic 1	project	use	build	version	error	work	file	tri	app	run	studio	xcode
Topic 2	px	width	height	color	style	background	div	left	top	font	var	text
Topic 3	input	use	imag	tf	data	size	group	would	like	get	set	tri
Topic 4	system	net	file	use	microsoft	run	api	server	web	instal	error	version
Topic 5	id	from	angular	this	component	import	select	compon	router	export	users	error
Topic 6	array	data	use	key	like	would	function	string	need	way	const	read
Topic 7	div	class	item	button	li	menu	span	text	click	element	display	option

Topics / Mots

Docs / Topics

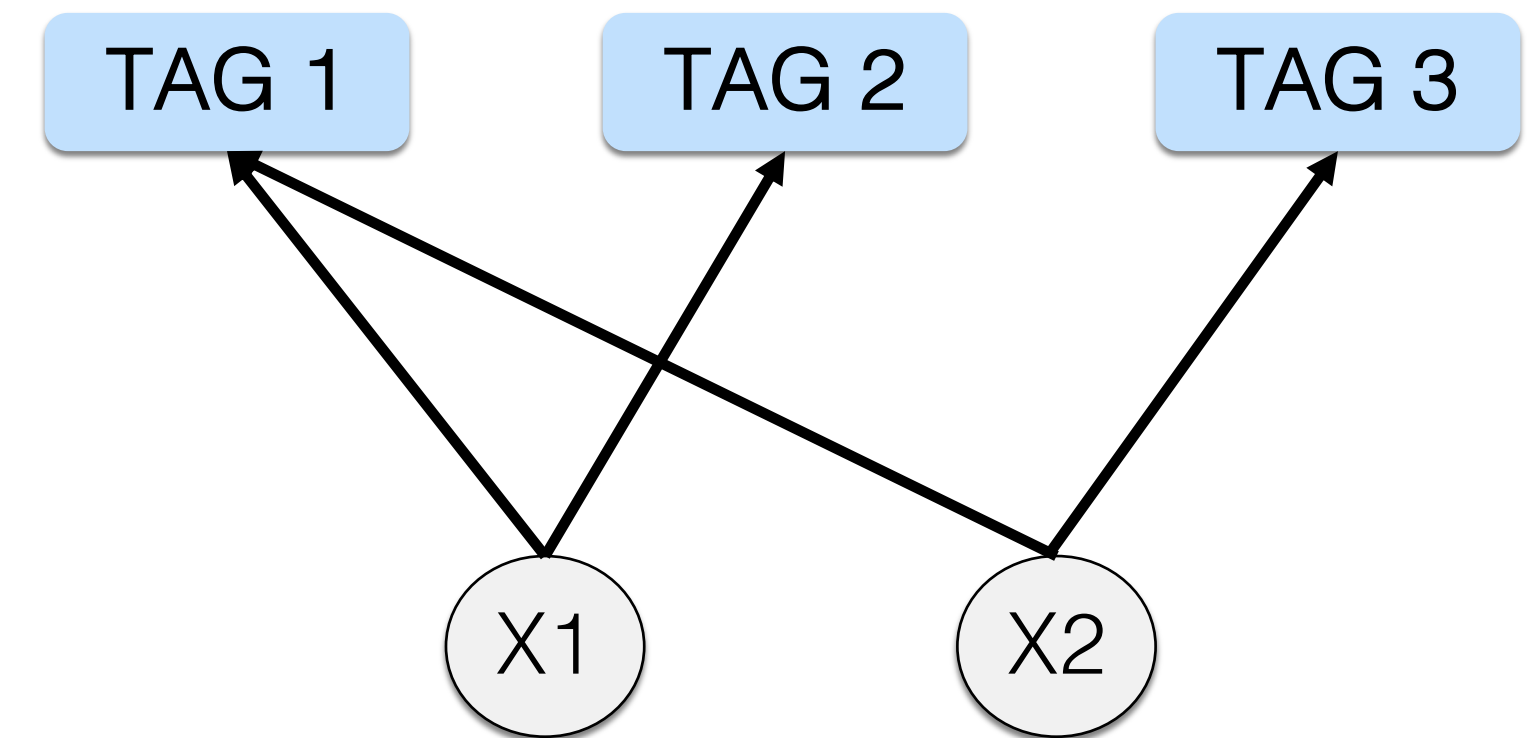
	Topic0	Topic1	Topic2	Topic3	Topic4	Topic5	Topic6	Topic7	Topic8	Topic9
Doc0	0.0025	0.9525	0.0025	0.0025	0.0025	0.0025	0.0025	0.0025	0.0025	0.0025
Doc1	0.0016	0.3394	0.0016	0.0016	0.0016	0.0016	0.0016	0.0016	0.0016	0.0016
Doc2	0.0002	0.0002	0.0538	0.0002	0.0002	0.0002	0.2236	0.0002	0.3213	0.0002
Doc3	0.0001	0.0882	0.0001	0.0001	0.0001	0.0001	0.7774	0.0001	0.0001	0.0312
Doc4	0.0015	0.4211	0.0015	0.0015	0.0015	0.0015	0.5524	0.0015	0.0015	0.0015
Doc5	0.0010	0.0010	0.0010	0.0010	0.2195	0.0010	0.0010	0.0010	0.0010	0.0010

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)

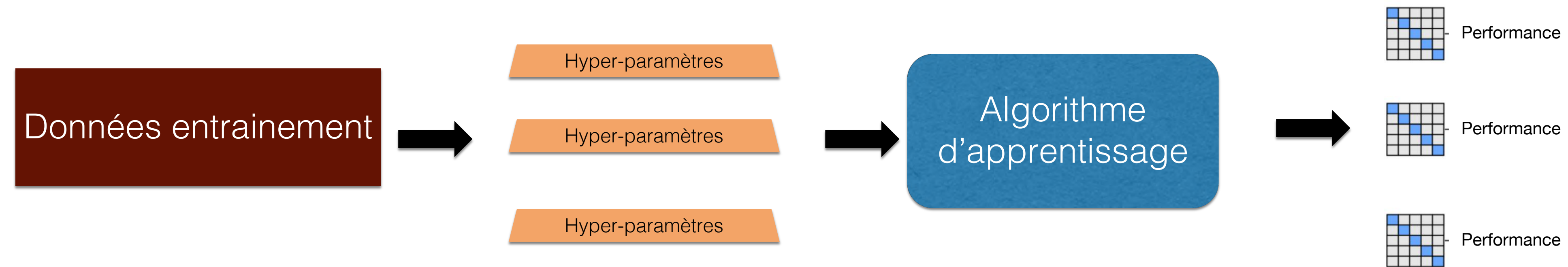
Gaussian Naives Bayes

Arbre de décision

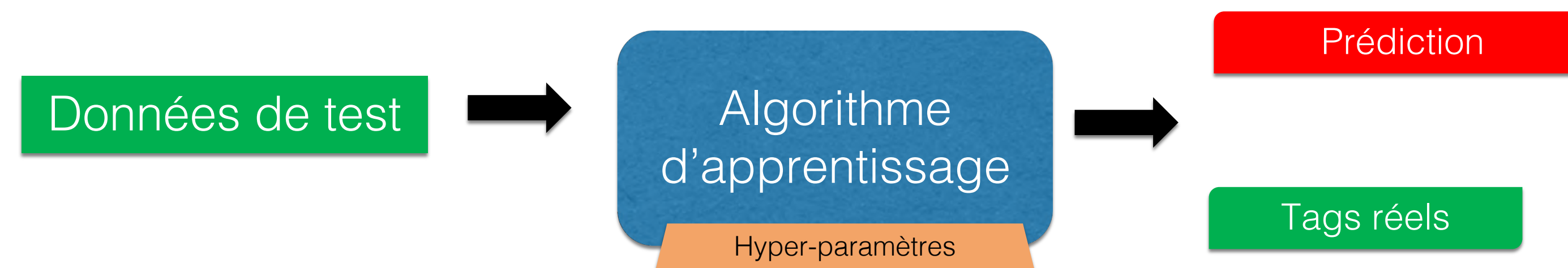
Forêt Aléatoire

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



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)

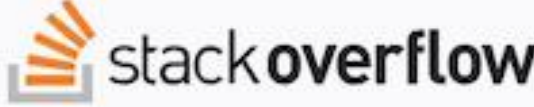
Tags

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