PROJET 5

Catégorisez automatiquement des questions

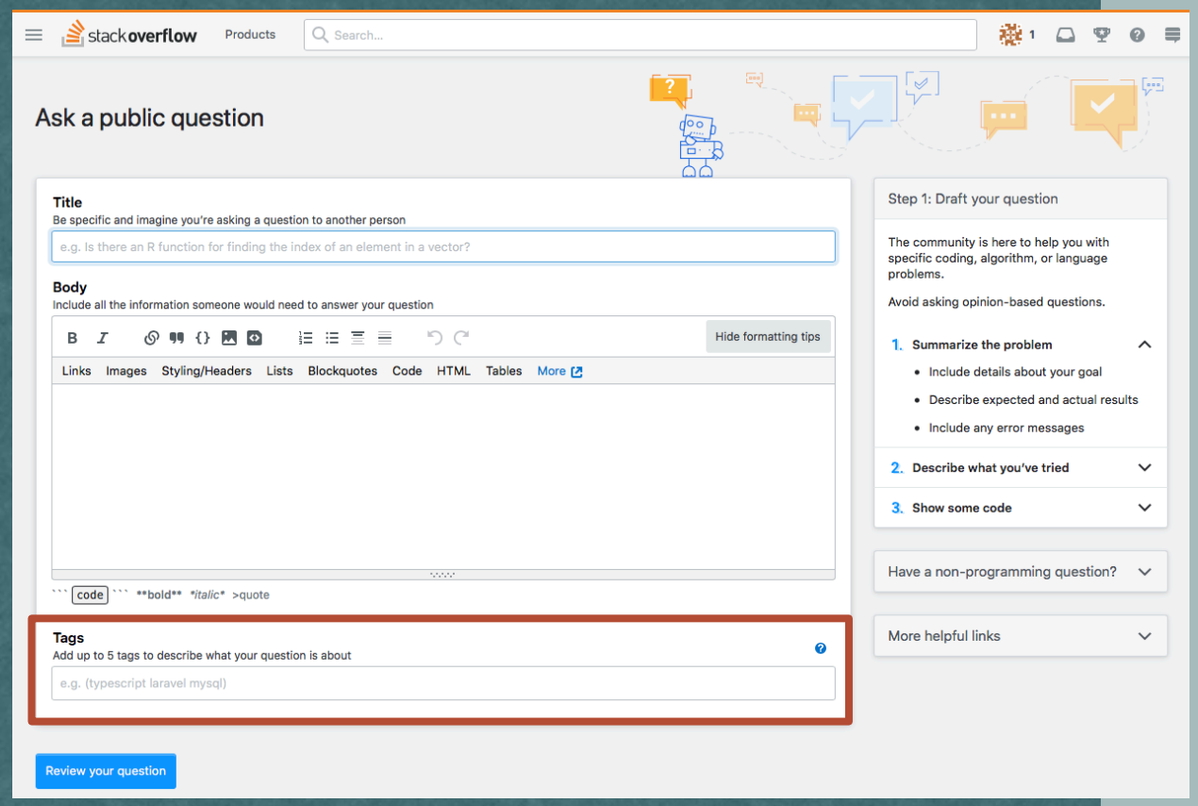
Marie-France LAROCHE-BARTHET

10/02/2022

INTRODUCTION

PROBLÉMATIQUE

- Site web **Stack Overflow**
  - Créé en 2008
  - Questions-réponses liées au développement informatique
  - Site de référence
- CONTEXTE du projet :
  - Entrer plusieurs tags
    - ➡ Facile pour utilisateur expérimenté
    - ➡ **Plus difficile** pour **nouvel** utilisateur
- BUT du projet :
  - **Système de suggestion de tags**
    - ➡ Algorithme de Machine Learning
    - ➡ Approche **non supervisée et supervisée**

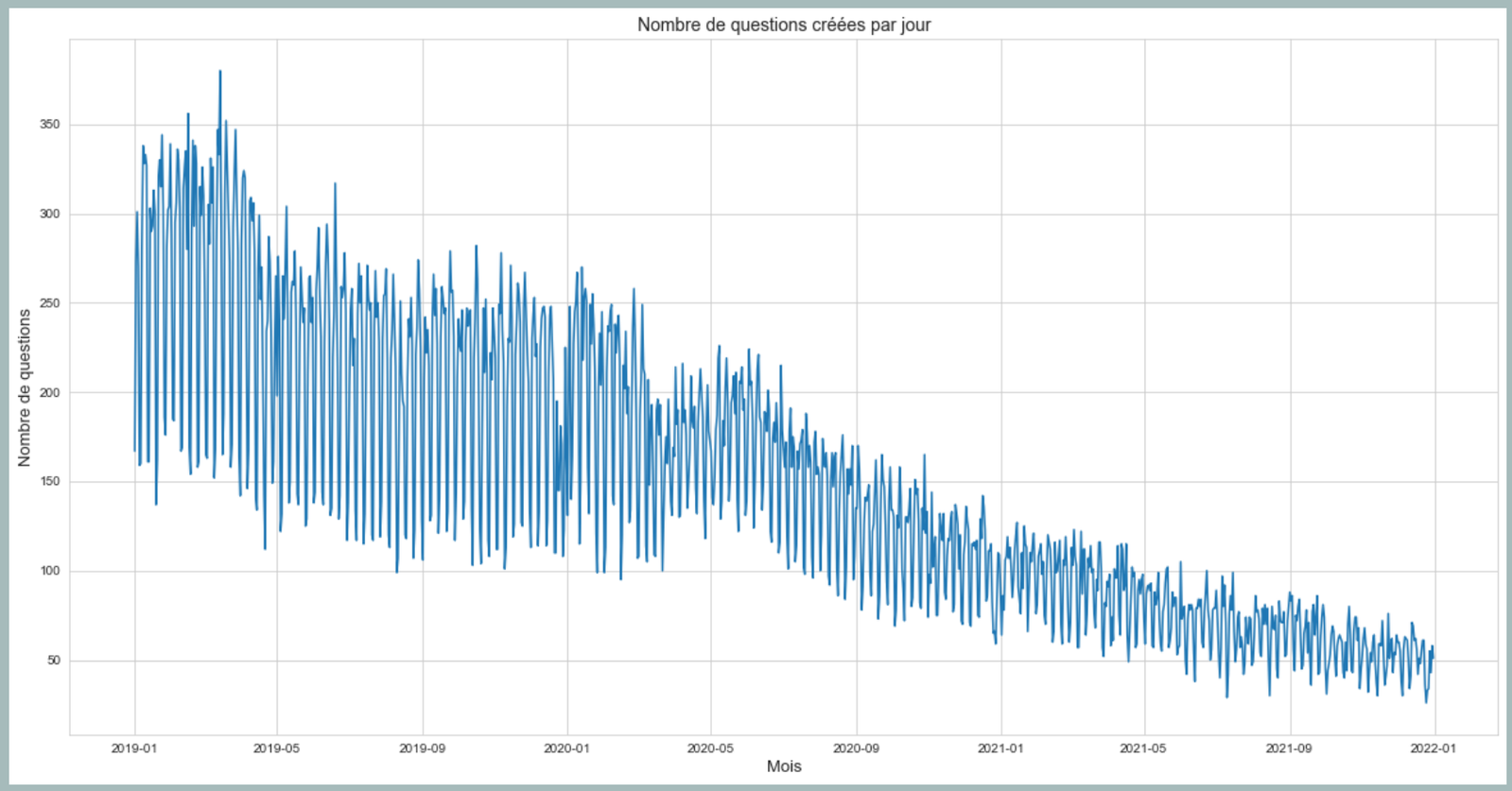


NETTOYAGE/ EXPLORATION DES DONNÉES

RÉCUPÉRATION DES DONNÉES

- Base de données **StackExchange Explorer**
  - Requêtes SQL
  - Limitées à 50000 lignes
- Récupération **(164535 x 9)**
  - **Questions** posées entre 2019 et 2021
  - Score, ViewCount, AnswerCount, CommentCount et FavoriteCount >= 1
- Choix d'un filtre
  - Evolution du nombre de questions créées au cours du temps
    - ➡ Diminution
    - ➡ Favorise les anciens posts
    - ➡ Autre critère de choix

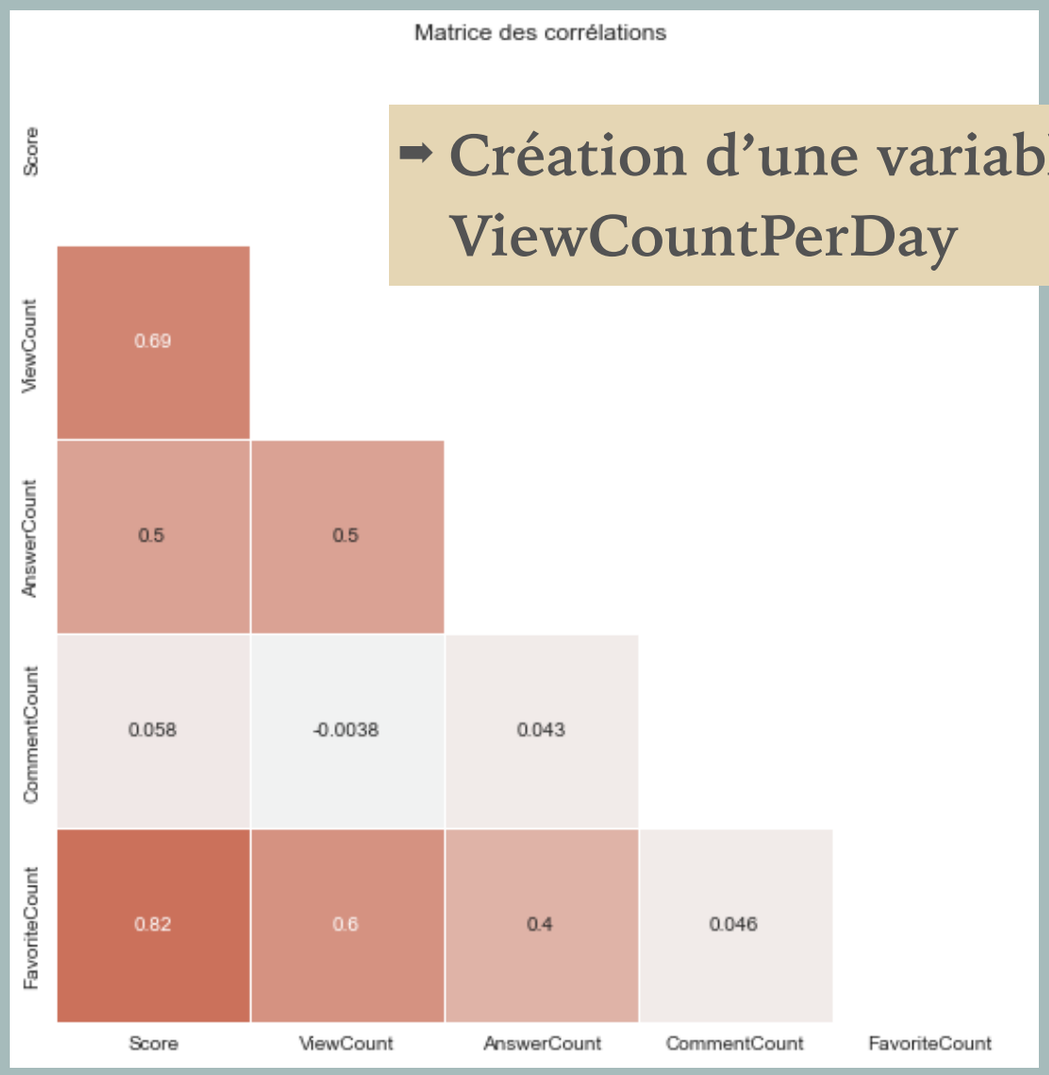
```
SELECT Id, Title, Body, Tags, CreationDate, Score, ViewCount, AnswerCount,
CommentCount, FavoriteCount
From Posts
WHERE PostTypeId = 1
AND CreationDate BETWEEN '2020-01-01T00:00:00.00' AND '2020-06-30T23:59:59.999'
AND Score >= 1
AND ViewCount >=1
AND AnswerCount >=1
AND CommentCount >=1
AND FavoriteCount >=1
ORDER BY CreationDate
```



EXPLORATION DES DONNÉES – CHOIX D'UN FILTRE

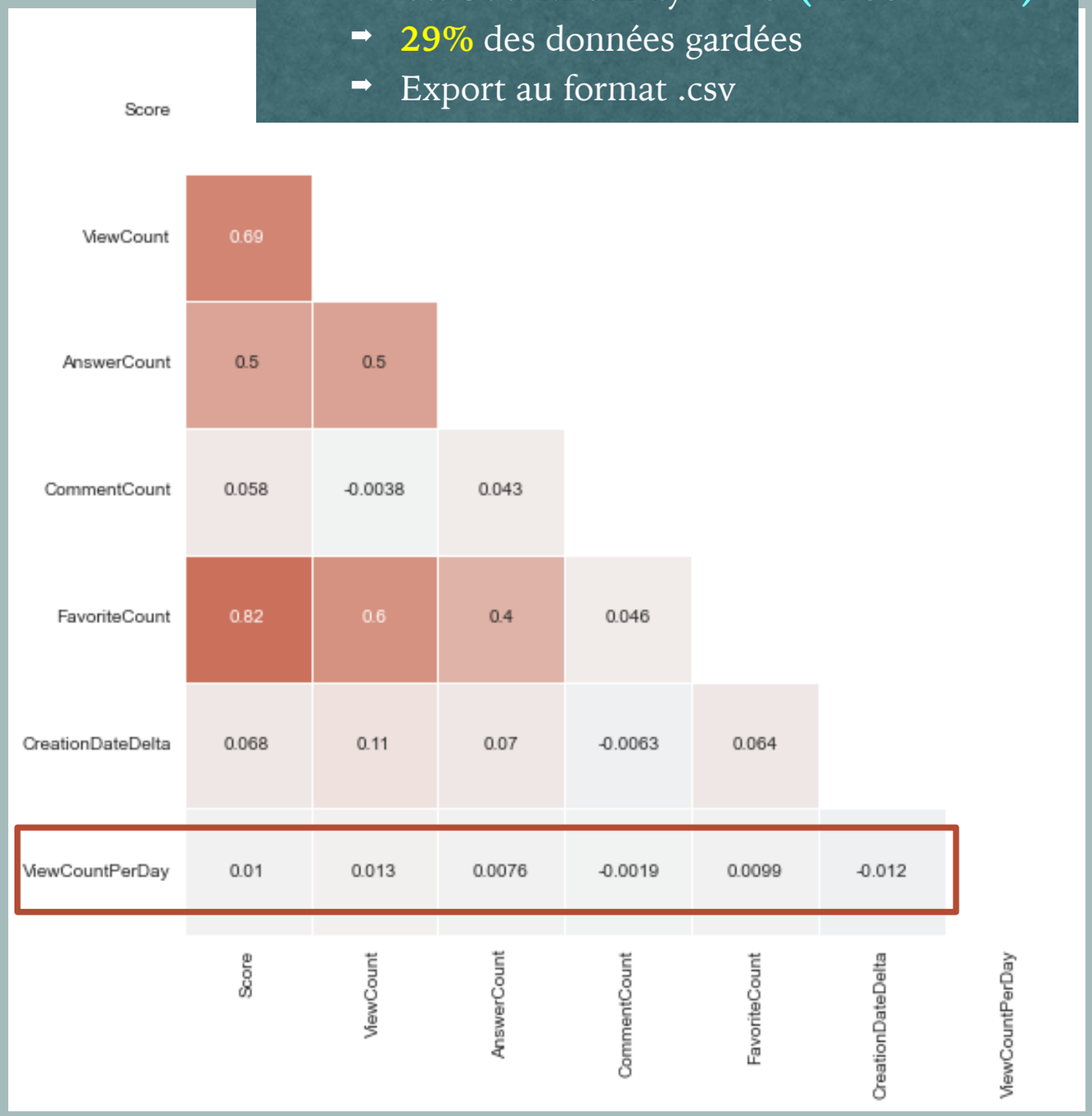
➤ *Distribution bivariée des variables*

➤ Matrice des corrélations
- Variables liées
  - ➡ CommentCount peu liée mais peu pertinente



➡ **Création d'une variable ViewCountPerDay**

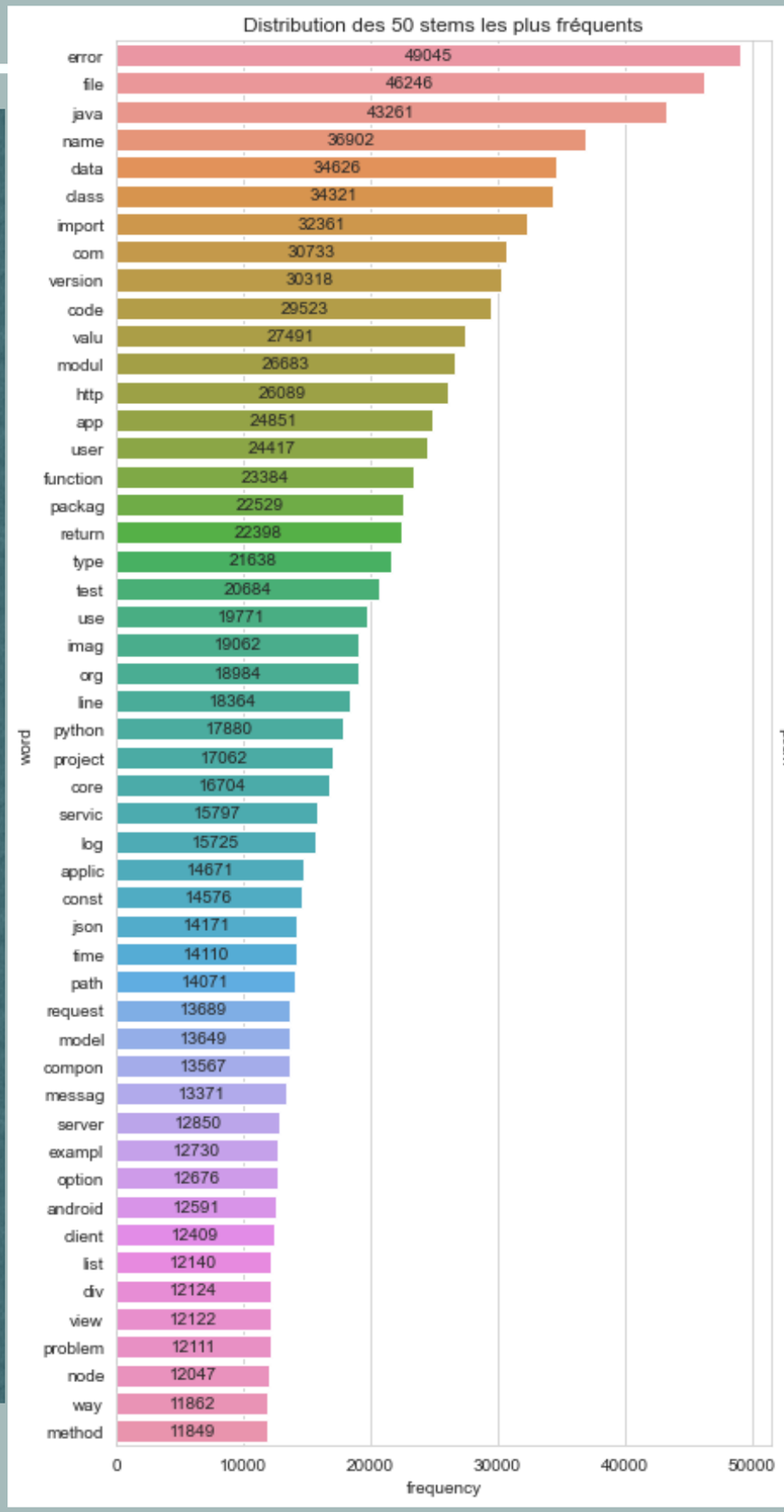
➤ Filtrage
- ViewCountPerDay >=3 **(47032 x 11)**
  - ➡ **29%** des données gardées
  - ➡ Export au format .csv

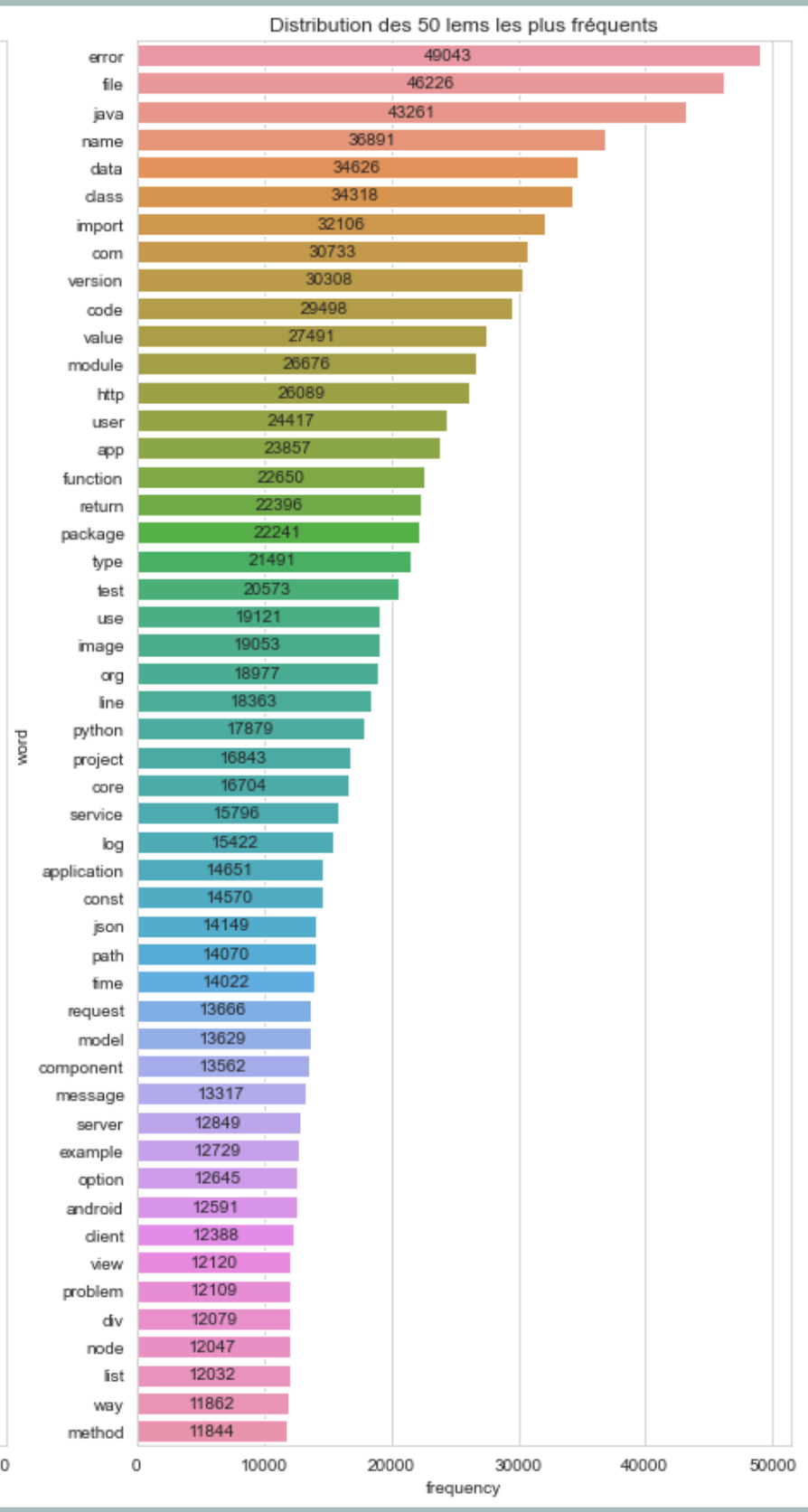


PRÉ-TRAITEMENTS EFFECTUÉES

PRÉ TRAITEMENTS

- Colonnes gardées (47032 x 3)
  - **Title, Body, Tags**
- Traitement Title+Body
  - Fusion Title+Body
  - Suppression balises HTML
  - Récupération des mots avec des lettres (**au moins 3 lettres**)
  - Mots mis en minuscule
  - Normalisation des données
    - ➡ **lemmatisation**
    - ➡ **stemmatisation**
- Distribution des stems et des lems :
  - 13 premiers mots identiques
  - Ordre changé par la suite mais mêmes 50 premiers mots



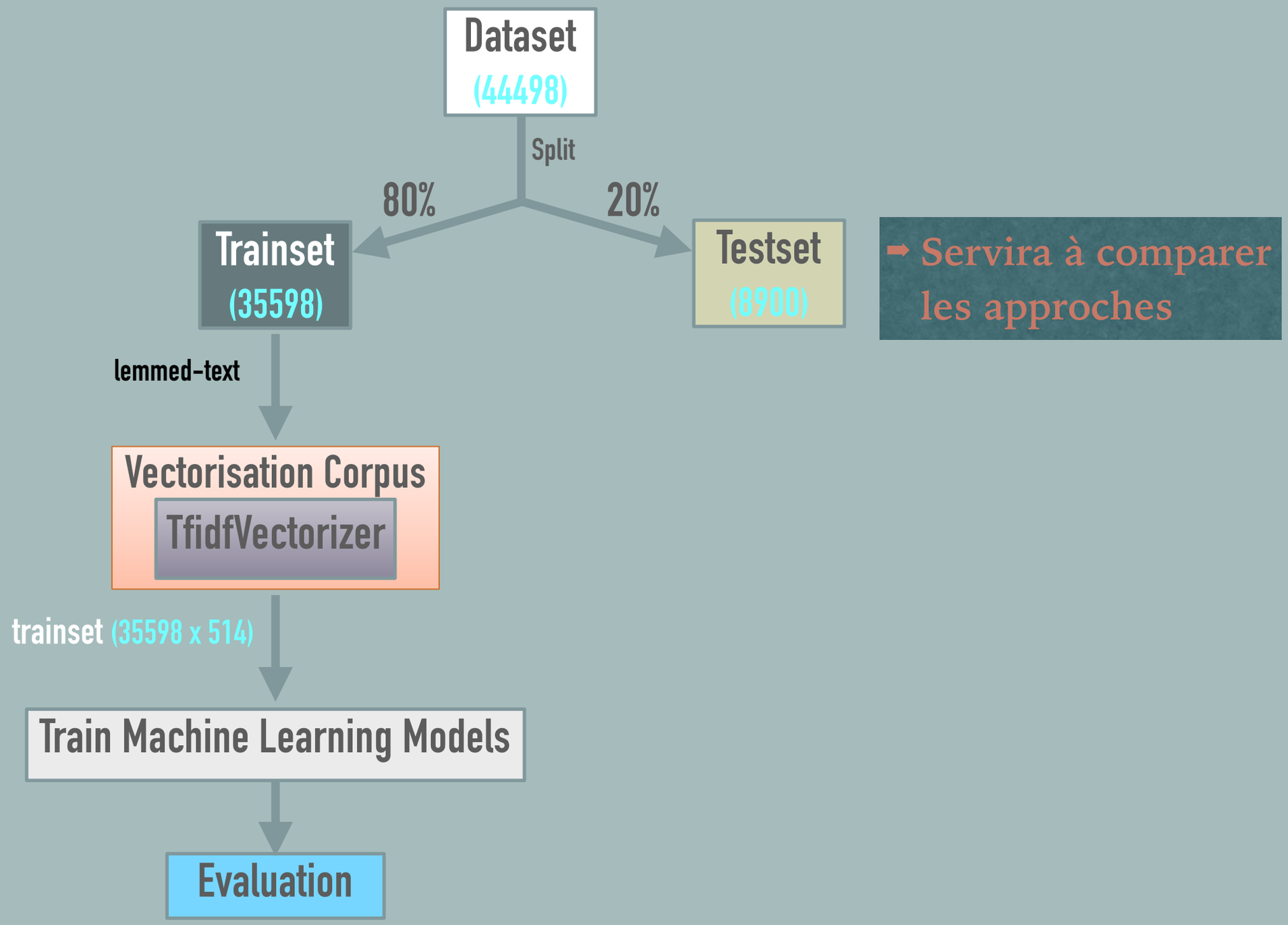


PRÉ TRAITEMENTS

- Traitement identique pour Tags
- Suppression de lignes avec listes vides (46806 x 3) 0.5%
  - Tags nettoyés vides
- Conservation des **lems**
  - Sémantique plus riche
- **Filtrage sur les 200 premiers lems tags** (44498 x 11) 4.9%
  - Fréquence d'au moins 150 (environ)
  - Ne pas surcharger l'apprentissage supervisé
  - Apparition de lignes sans tags
    - ➡ Suppression de ces lignes
  - Sauvegarde du dataset
- Récupération des **lems** ayant une occurence d'au moins 1400
  - **514 mots**
  - Ne pas surcharger la vectorisation
  - Sauvegarde de la liste

MODÉLISATIONS EFFECTUÉES

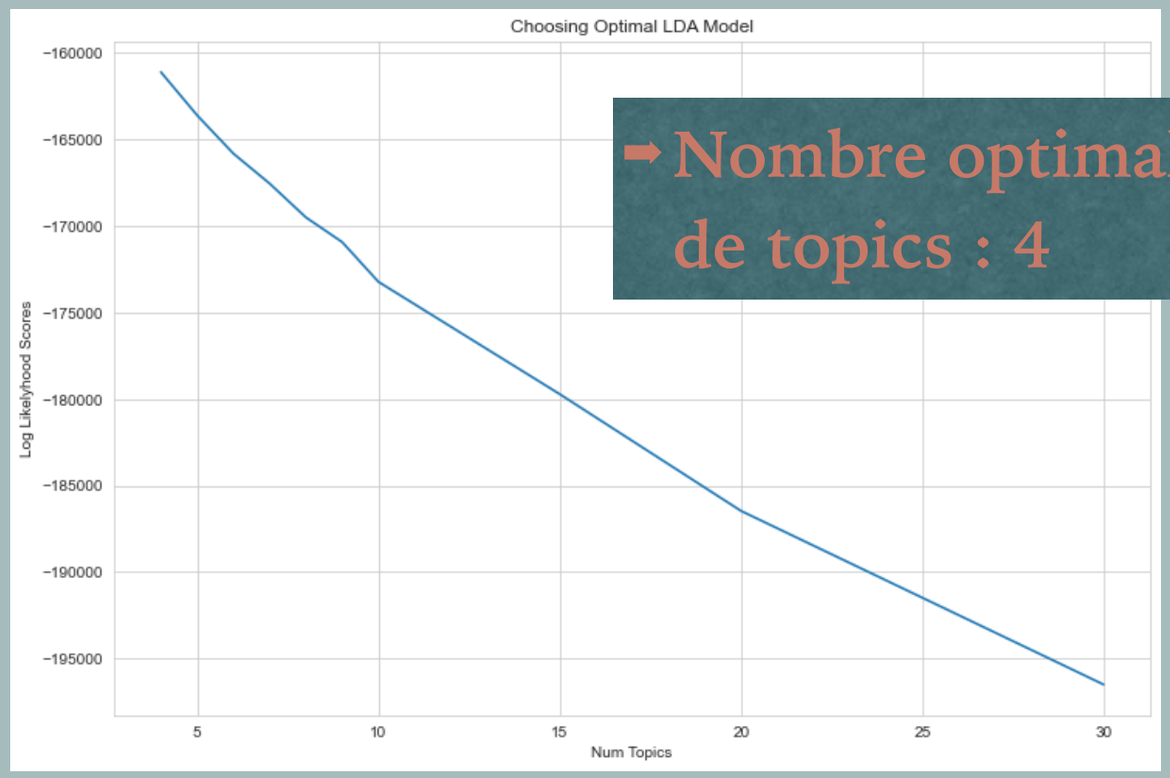
TRONC COMMUN APPROCHES NON-SUPERVISÉE / SUPERVISÉE



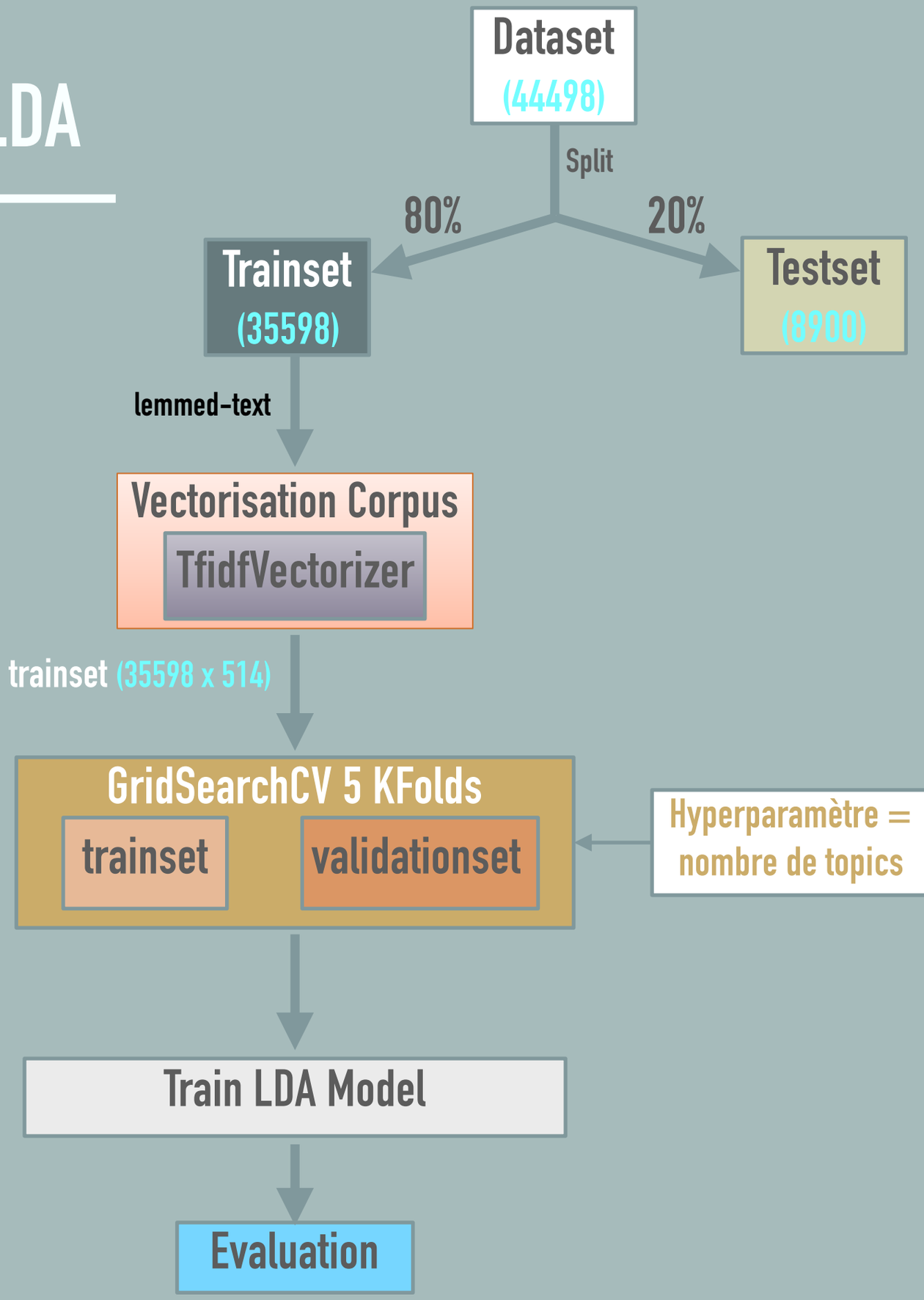
APPROCHE NON SUPERVISÉE – LDA

➤ *Choix du nombre de topics*

- ➤ **Nombre de topics testés**
  - 4,5,6,7,8,9,10,15,20,30
- ➤ **Métrique utilisée**
  - Maximum de vraisemblance log likelihood

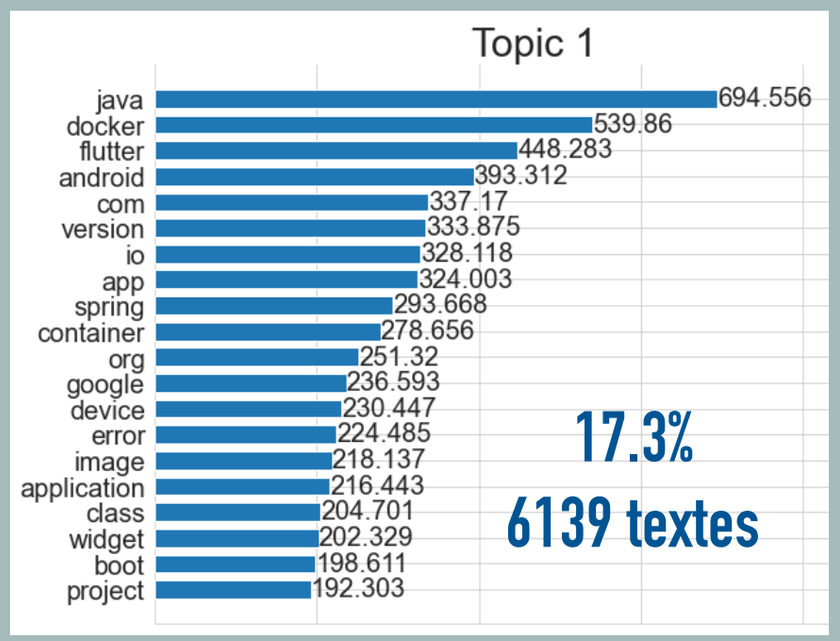


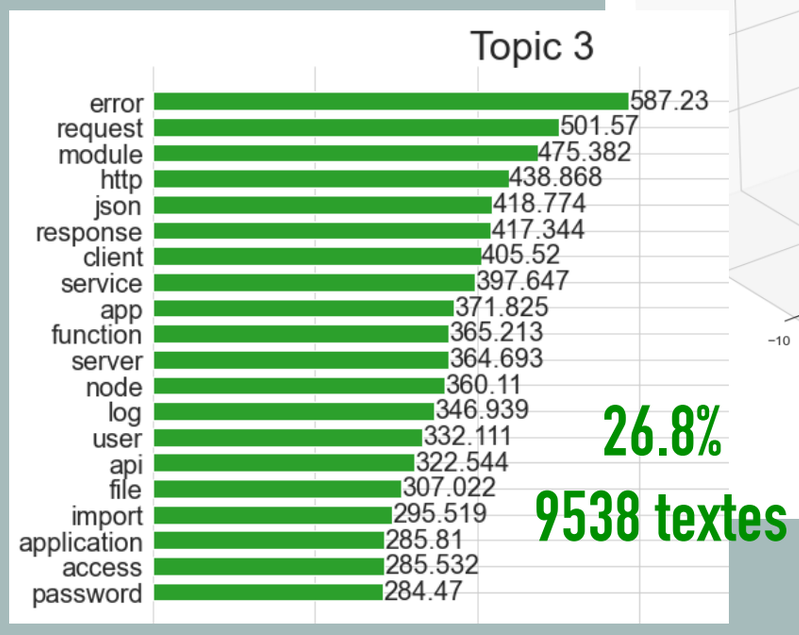
➡ Nombre optimal de topics : 4

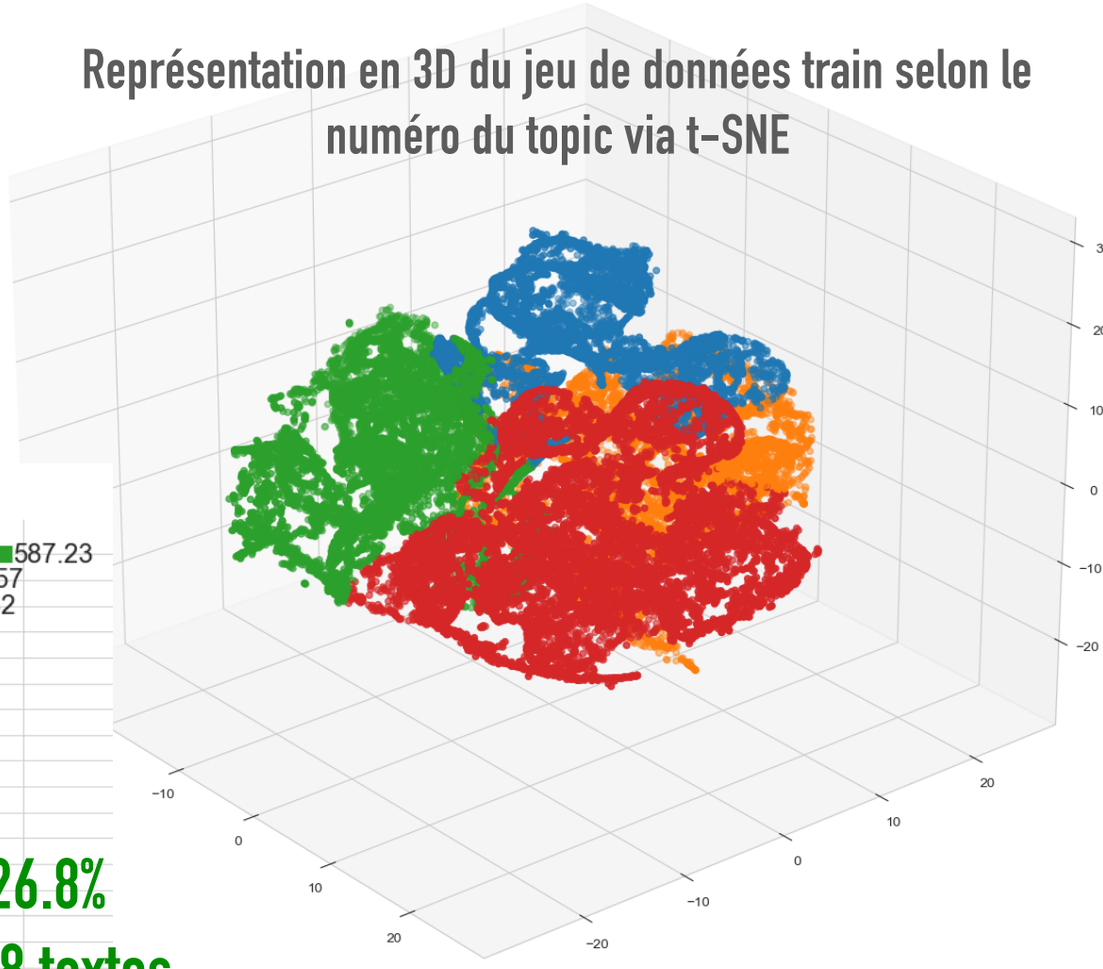


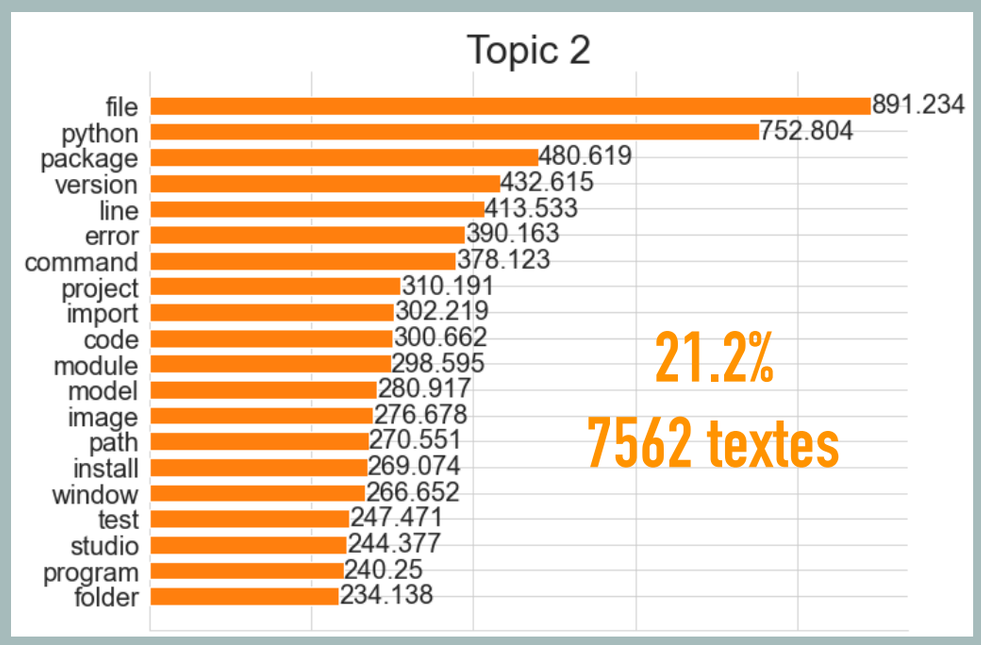
APPROCHE NON SUPERVISÉE – LDA

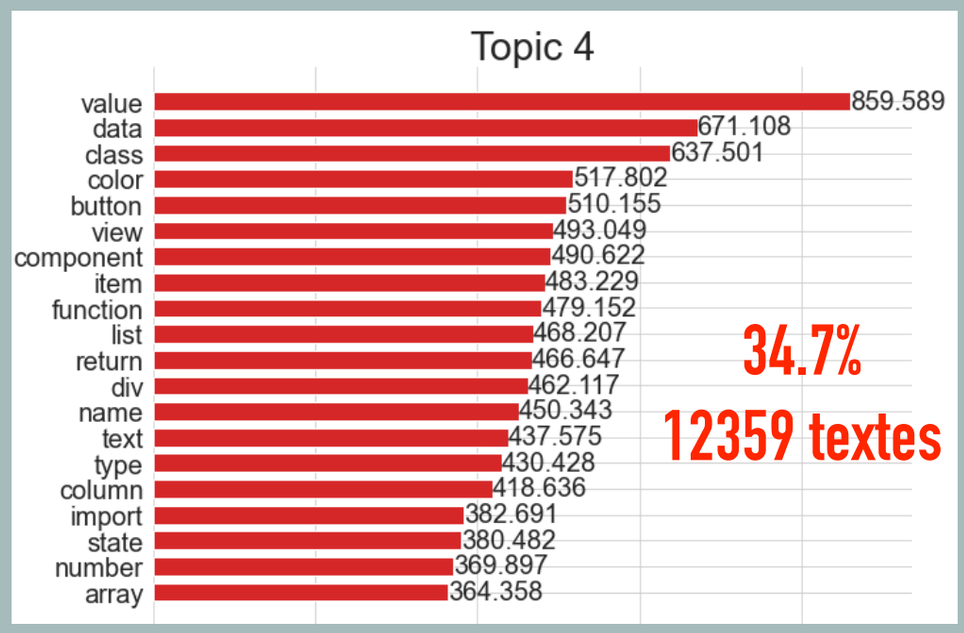
➤ *Distribution des topics*







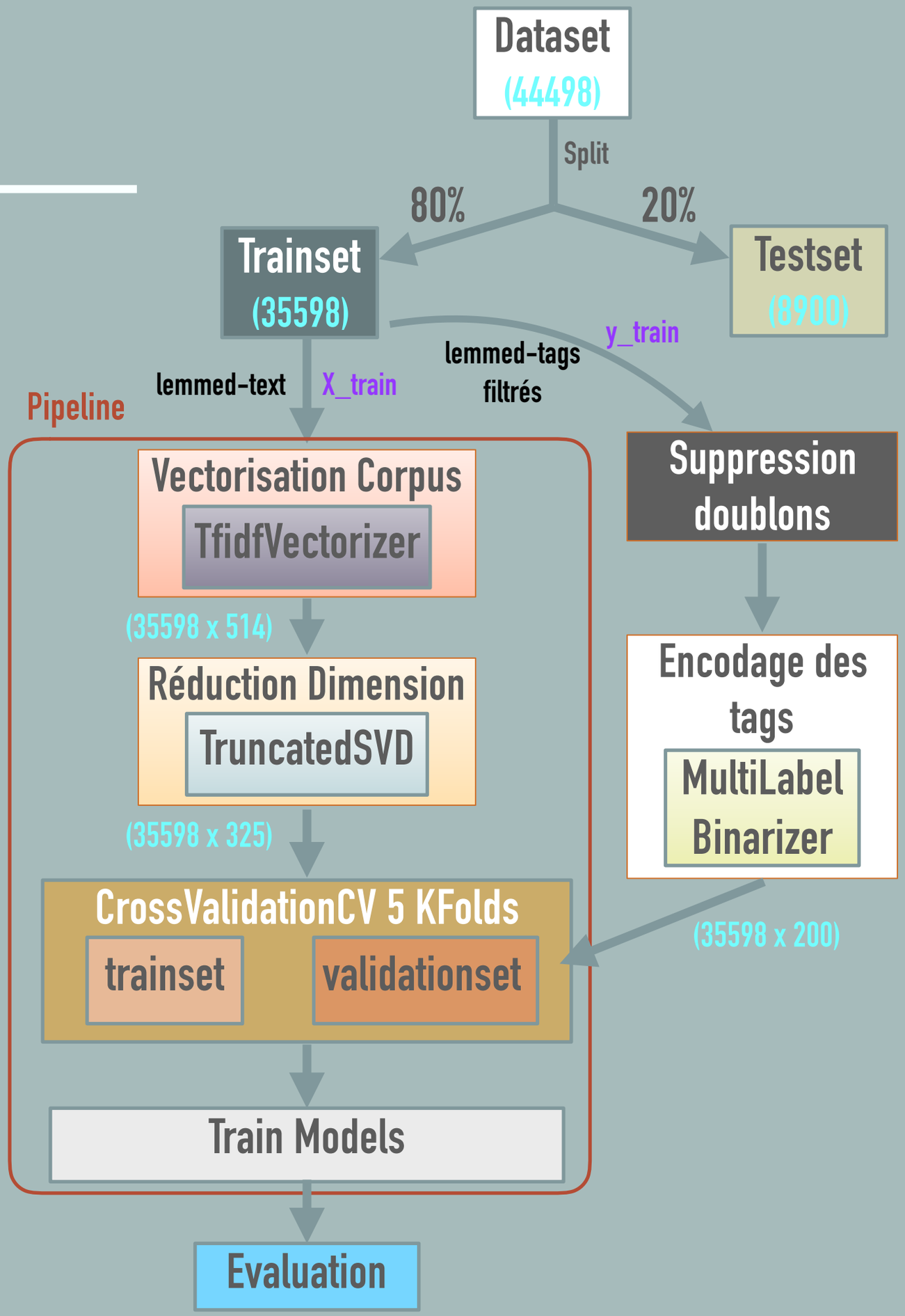




- ➤ 2 dictionnaires pour stocker premiers mots de chaque topic
  - 20 mots
  - 50 mots
- ➡ Pour analyse avec approche supervisée

APPROCHE SUPERVISÉE

- **Modèles testés**
  - DummyClassifier (Dummy)
  - KNeighborsClassifier (KNN)
  - LinearSVC couplé à OneVsRestClassifier (SVM)
  - RandomForestClassifier (Random Forest)
- **Valeurs par défaut** des hyperparamètres
- Métriques utilisées
  - **Accuracy**
  - **Precision**
  - **Recall**
  - **F1**



APPROCHE SUPERVISÉE

➤ *Résultats*

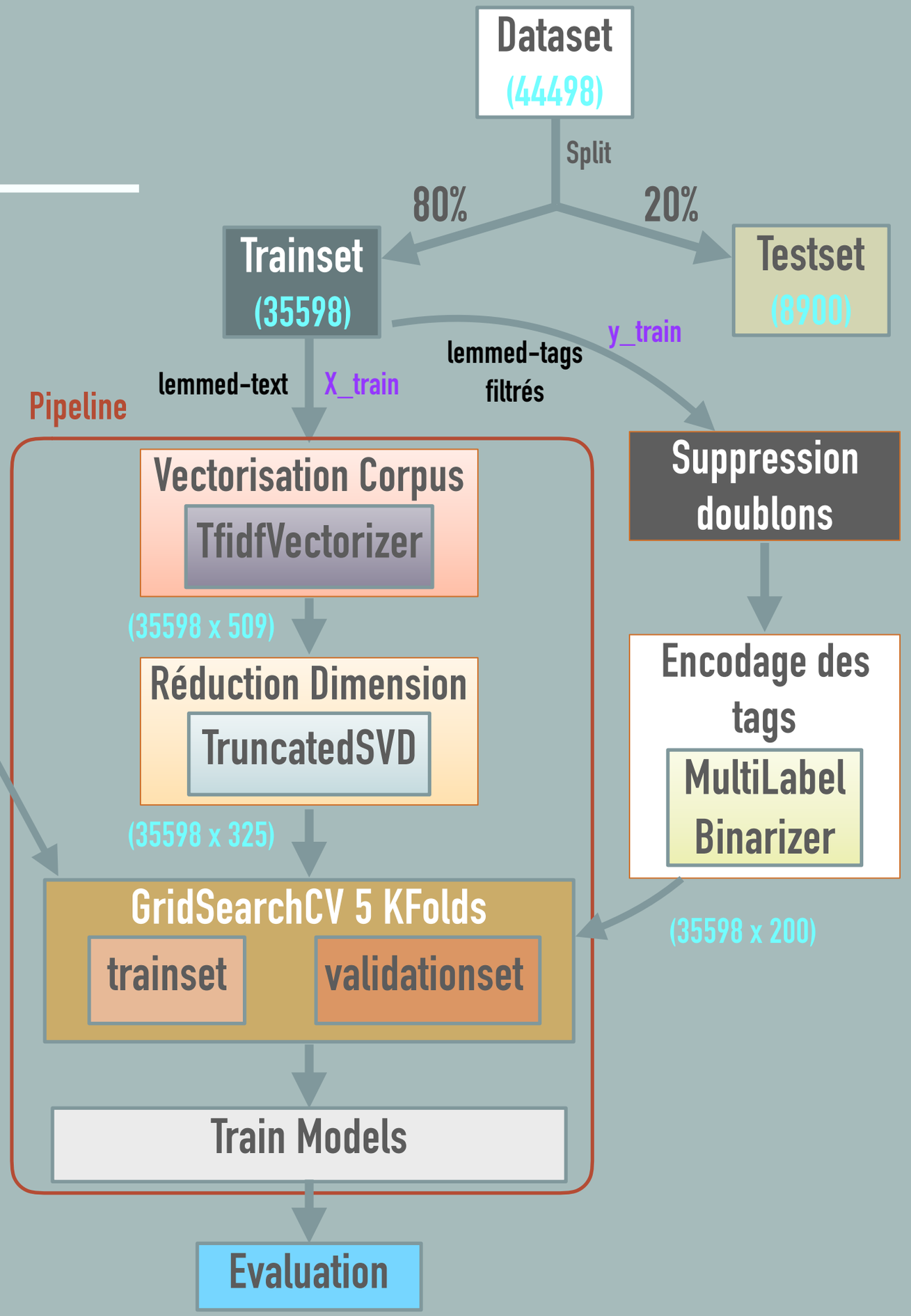
Modèle	Accuracy	Precision micro	Recall micro	f1 micro	Temps d'entrainement
Dummy	0.0	0.013115	0.498786	0.025558	37.1s
KNN	0.100483	0.688424	0.273560	0.391530	3min 58s
SVM	0.147256	0.800425	0.367456	0.503668	3min 48s
Random Forest	0.048851	0.888180	0.115611	0.204583	48min 35s

➤Modèle retenu

- **SVM**

APPROCHE SUPERVISÉE

➤ *Optimisation du SVM*



Hyperparamètre à tester = C

C	Accuracy	Precision micro	Recall micro	f1 micro
0.001	0.003849	0.947473	0.006432	0.012777
0.01	0.063318	0.873285	0.154423	0.262435
0.1	0.122957	0.823625	0.304882	0.445006
1	0.147256	0.800425	0.367456	0.503668
10	0.149812	0.768308	0.393184	0.520163

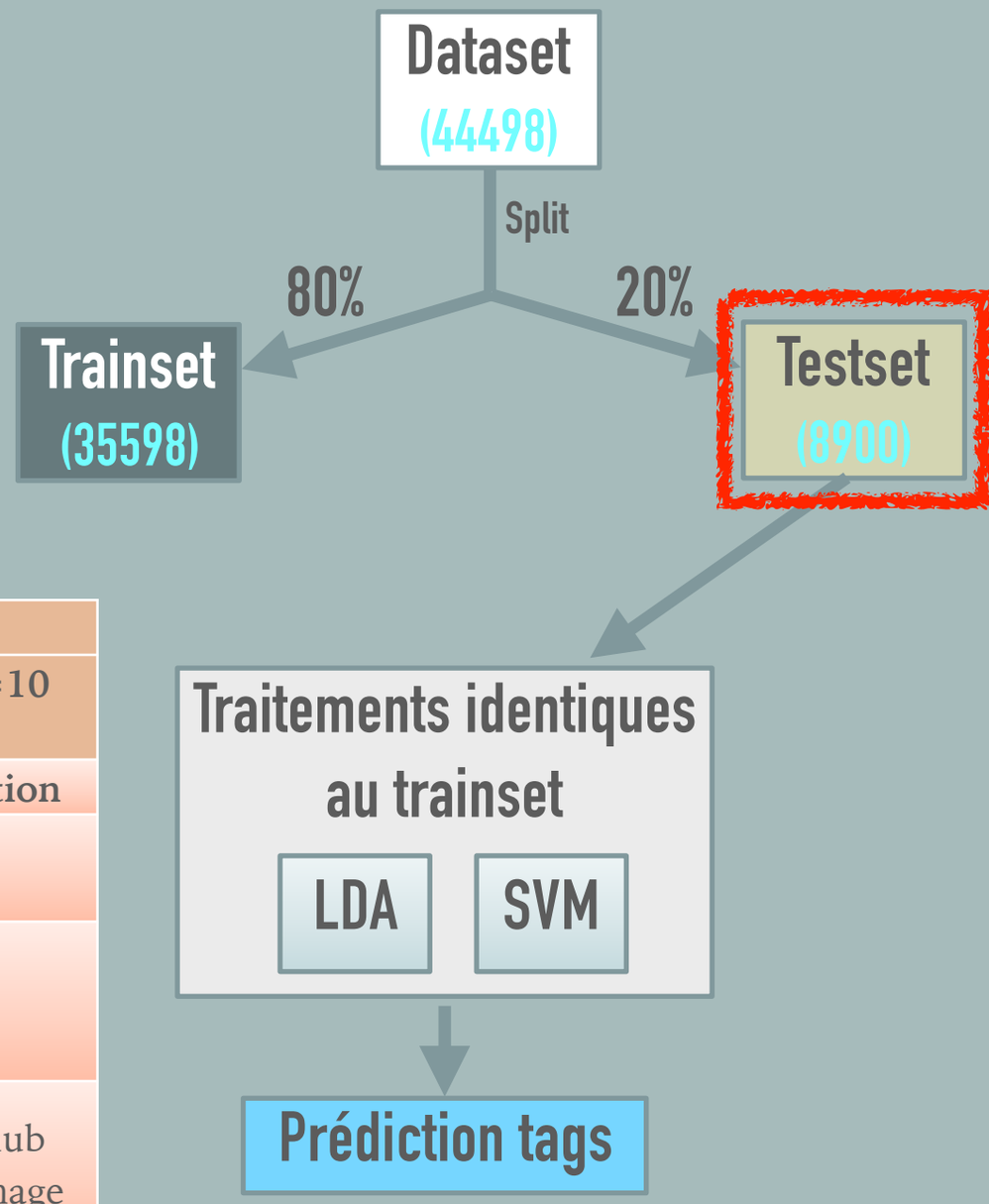
➤Deux modèles conservés pour comparaison finale

- C =1
- C = 10

COMPARAISON DES APPROCHES

➤ *Qualitativement*

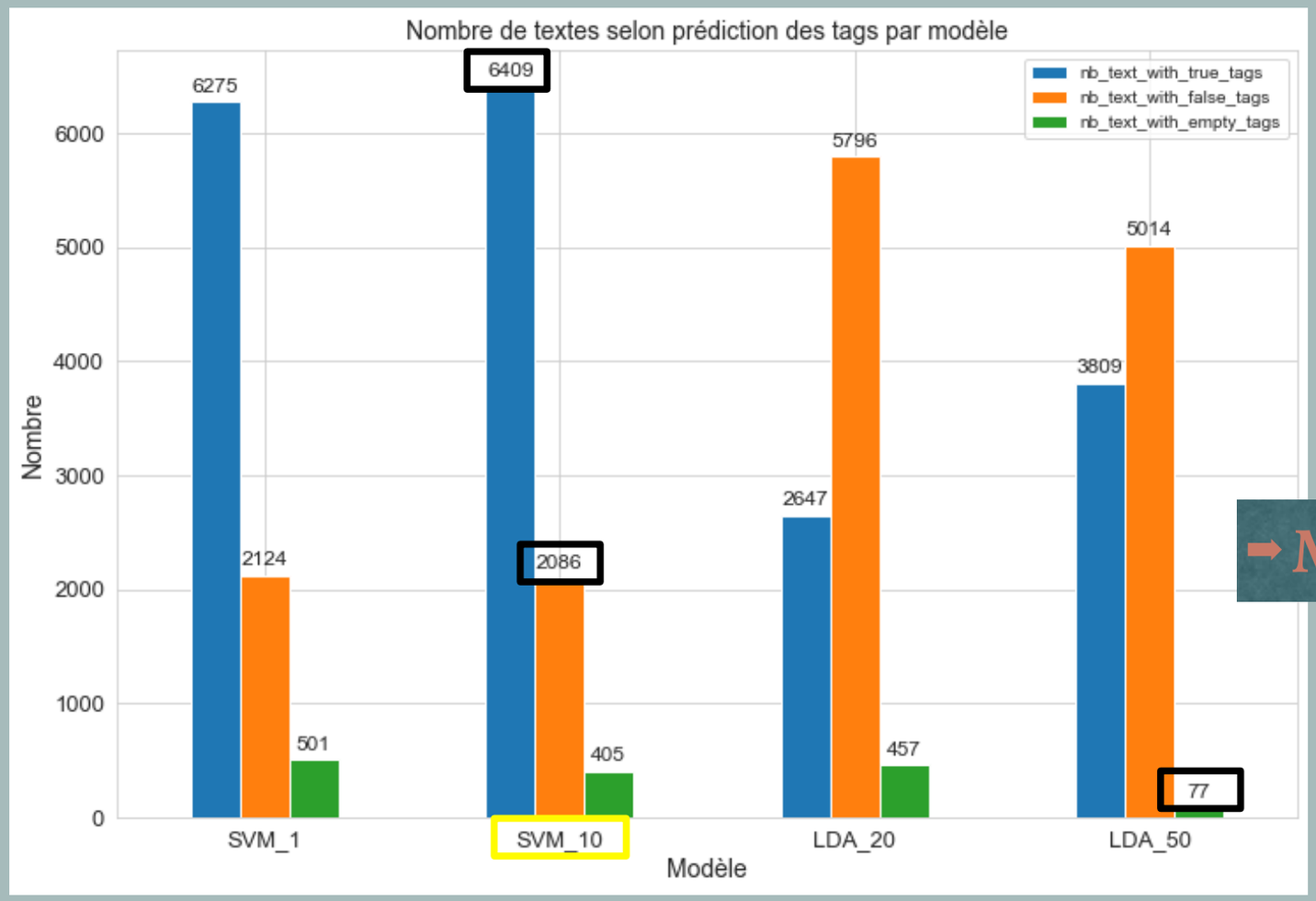
Tags utilisateurs (lems tags)	Approche non supervisée		Approche supervisée	
	LDA 20 mots	LDA 50 mots	SVM C=1	SVM C=10
github action	project test	project test job way run	server **action**	server **action**
reactjs react hook	-	context	**hook**	**hook**
android viewmodel mvvm	value data class button view return name type	value data class button view return name type page	-	type
docker compose codespaces	**docker** container image	**docker** container image build test volume	json github **docker** image	json github **docker** image
php laravel vue pusher	error http app server	error http app server use com **php** post route option name config login console	server **laravel** **php** image	server **laravel** **php** http image

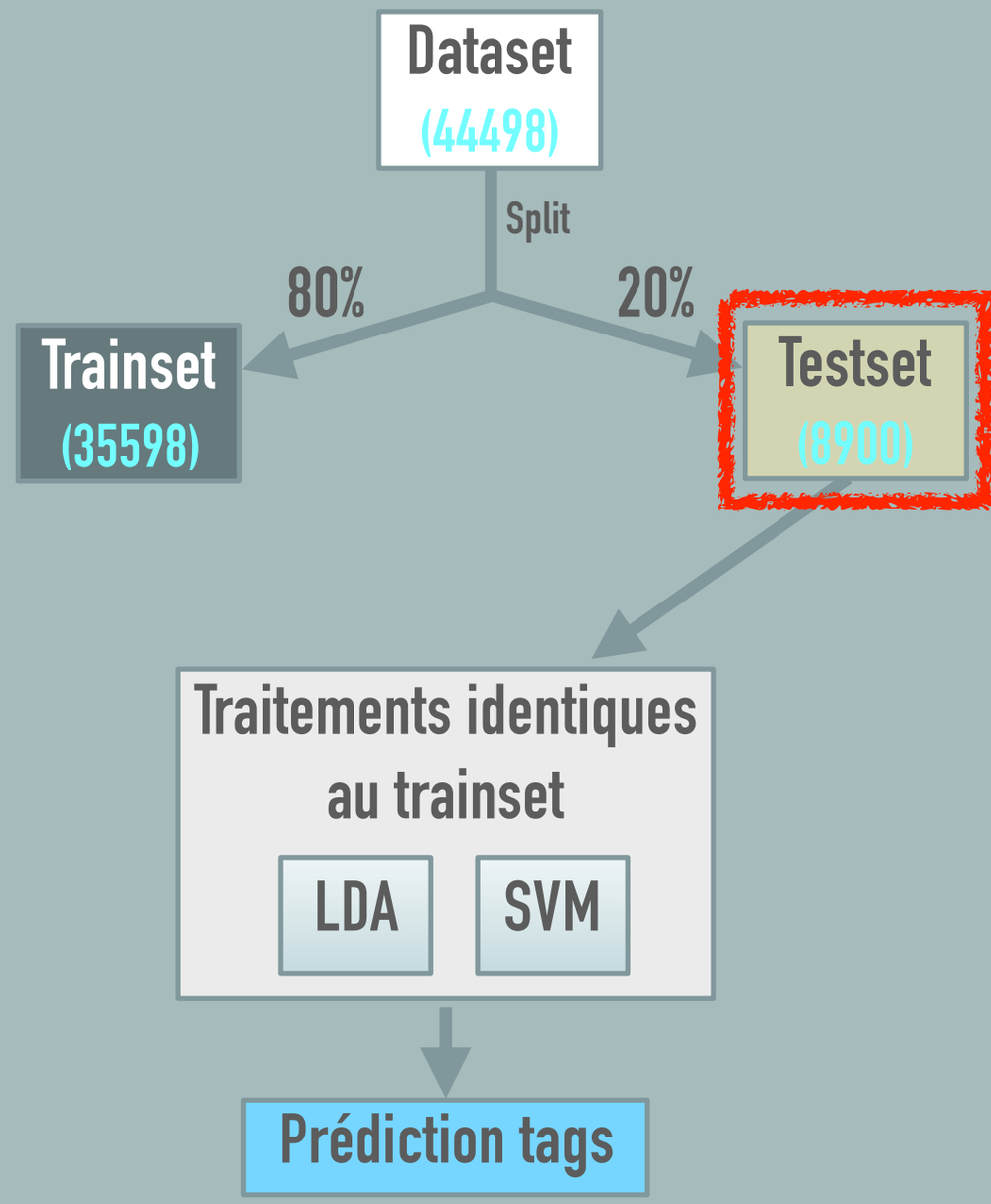


COMPARAISON DES APPROCHES

➤ *« Quantitativement »*

➤ **Pour chaque texte, si modèle prédit :**
- Aucun tag, score = -1 (« **empty tags** »)
- Aucun tag **dans liste lems tags**, score = 0 (« **false tags** »)
- **Au moins un tag** dans liste lems tags, score = 1 (« **true tags** »)





➡ Modèle retenu SVM C =10

MODÈLE FINAL

MODÈLE SÉLECTIONNÉ

- **SVM**
  - Intégré dans **pipeline** contenant
    - Vectorisation Corpus : TfidfVectorizer
    - Réduction de dimension : TruncatedSVD
    - Classifieur : LinearSVC(C = 10) couplé à OneVsRestClassifier
  - Sauvegardé au format .joblib
- **Encodage des tags**
  - MultiLabelBinarizer
  - Sauvegardé au format .joblib
- Mis dans API
  - FastApi

API

Tags Prediction Application - S

127.0.0.1:8000/docs#/

Applications Gmail YouTube Maps Liste de lecture

Tags Prediction Application 0.0.1 OAS3

/openapi.json

Application to give tags prediction for question with FastAPI + uvicorn

default

GET / Root

POST /predict Get Prediction

Schemas

HTTPValidationError

Input

ValidationError

Tags Prediction Application - S

127.0.0.1:8000/docs#/default/get_prediction_predict_post

Applications Gmail YouTube Maps Liste de lecture

Tags Prediction Application 0.0.1 OAS3

/openapi.json

Application to give tags prediction for question with FastAPI + uvicorn

default

GET / Root

POST /predict Get Prediction

Parameters Try it out

No parameters

Request body required application/json

Example Value | Schema

```
{
  "question": "string",
  "description": "string"
}
```

Tags Prediction Application - S

127.0.0.1:8000/docs#/default/get_prediction_predict_post

Applications Gmail YouTube Maps Liste de lectu

Request URL

```
http://127.0.0.1:8000/predict
```

Server response

Code	Details
200	Response body

```
{
  "Enter your question": "How to show real time data from from realtime database?",
  "Explain your problem": "I'm making this realy simple project where when you click a button, it in
crements a value in the realtime databse. This works, but I want a Text widget to update everytime t
he value changes. I've really new to this so please excuse my lack of knowledge. I have noticed that
there's no snapshots methods available that would let me do something like this, so how I update a l
abel to be the data in the realtime storage eveyime it changes? Here's the code which I use to incre
ment the values-",
  "Tags prediction": "storage"
}
```

Download

Response headers

```
content-length: 625
content-type: application/json
date: Sun,06 Feb 2022 08:24:46 GMT
server: uvicorn
```

Tags Prediction Application 0.0.1 OAS3

/openapi.json

Application to give tags prediction for question with FastAPI + uvicorn

default

GET / Root

POST /predict Get Prediction

Parameters Cancel Reset

No parameters

Request body required application/json

```
{
  "question": "How to show real time data from from realtime database?",
  "description": "I'm making this realy simple project where when you click a button, it increments a value in
the realtime databse. This works, but I want a Text widget to update everytime the value changes. I've really
new to this so please excuse my lack of knowledge. I have noticed that there's no snapshots methods available
that would let me do something like this, so how I update a label to be the data in the realtime storage
eveyime it changes? Here's the code which I use to increment the values-"
}
```

Execute

CONCLUSION

CONCLUSION

- Modèle retenu pour prédiction de tags
  - **LinearSVC, supervisé**
- Difficulté de comparaison des approches non-supervisée et supervisée
  - Absence de métriques communes
- **Pistes d'amélioration :**
  - Traitement du texte
    - ➡ Création d'un dictionnaire spécifique pour les mots à conserver (langages de programmation, caractères spéciaux)
  - Plus d'échantillons (ici ~50 000)
  - Plus de variables (ici 514 premiers mots, 200 premiers tags)
  - Autres librairies que scikit-learn
    - ➡ Gensim pour LDA

MERCI

QUESTIONS