Ce rapport

présente les différents traitements effectués et détaille les modélisations effectuées.

Introduction

Stack Overflow est un site qui permet à ses utilisateurs de créer des questions en rapport avec le développement informatique. Lors de la création d'une question, il faut associer une ou plusieurs étiquettes qui décrivent le sujet.

Ces étiquettes sont des catégories ou classes à priori sans hiérarchie. Elles permettent un regroupement facile des questions lors d'une recherche ultérieure par l'utilisateur.

L'utilisateur doit réfléchir et trouver les étiquettes les plus pertinentes. C'est un effort cognitif supplémentaire désagréable pour l'expérience utilisateur.

Peut-on réaliser un système de suggestion d'étiquettes ?

Une solution envisagée est d'utiliser un algorithme d'apprentissage automatique pour suggérer des étiquettes à partir des données.

Nous avons deux classes d'algorithmes d'apprentissage automatique à expérimenter

- non supervisé (modélisation de sujets sans étiquettes)
- supervisé (classification avec étiquettes)

Expérimentation

Collecter les données

Stackoverflow propose une interface permettant d'effectuer des requêtes dans leur base de données qui s'appelle Stack Exchange Data Explorer

Lien de l'interface d'accès à la base de données :
https://data.stackexchange.com/stackoverflow/query/new

Nous allons récupérer 50000 rows au format .csv à l'aide de la requête SQL suivante :

```
SELECT Id,Body,Title,Tags
FROM posts
WHERE Id < 1000000
AND ParentId IS NULL
```

On identifie les champs à récupérer sur nos données disponibles lors de la création d'une question.

Ce sont les "Posts" qui n'ont pas de "ParentId".

On peut dépasser la limite de manière automatique, il est possible de récupérer directement dans un dataframe avec de multiples requêtes python sur l'api afin de dépasser la limite de 50 000 rows, puis d'exporter pour traitement et modélisation.

Nettoyer les données textuelles

Nous devons à partir de données textuelles obtenir un format compréhensible par nos algorithmes, c'est-à-dire réaliser un plongement de mots.

Il est donc nécessaire de réaliser un ensemble d'opérations de nettoyage au préalable afin d'identifier les tokens les plus importants par questions.

1. Importer et observer les données
2. Identifier et traiter les doublons
3. Identifier et traiter les valeurs manquantes

Body et Title

4. supprimer les balises html
5. convertir le type de casse à 'lower'
6. supprimer à l'aide d'une regex les chiffres et ponctuations
7. supprimer les mots courants anglais à l'aide de nltk
8. supprimer les mots inutiles identifiés

Tags

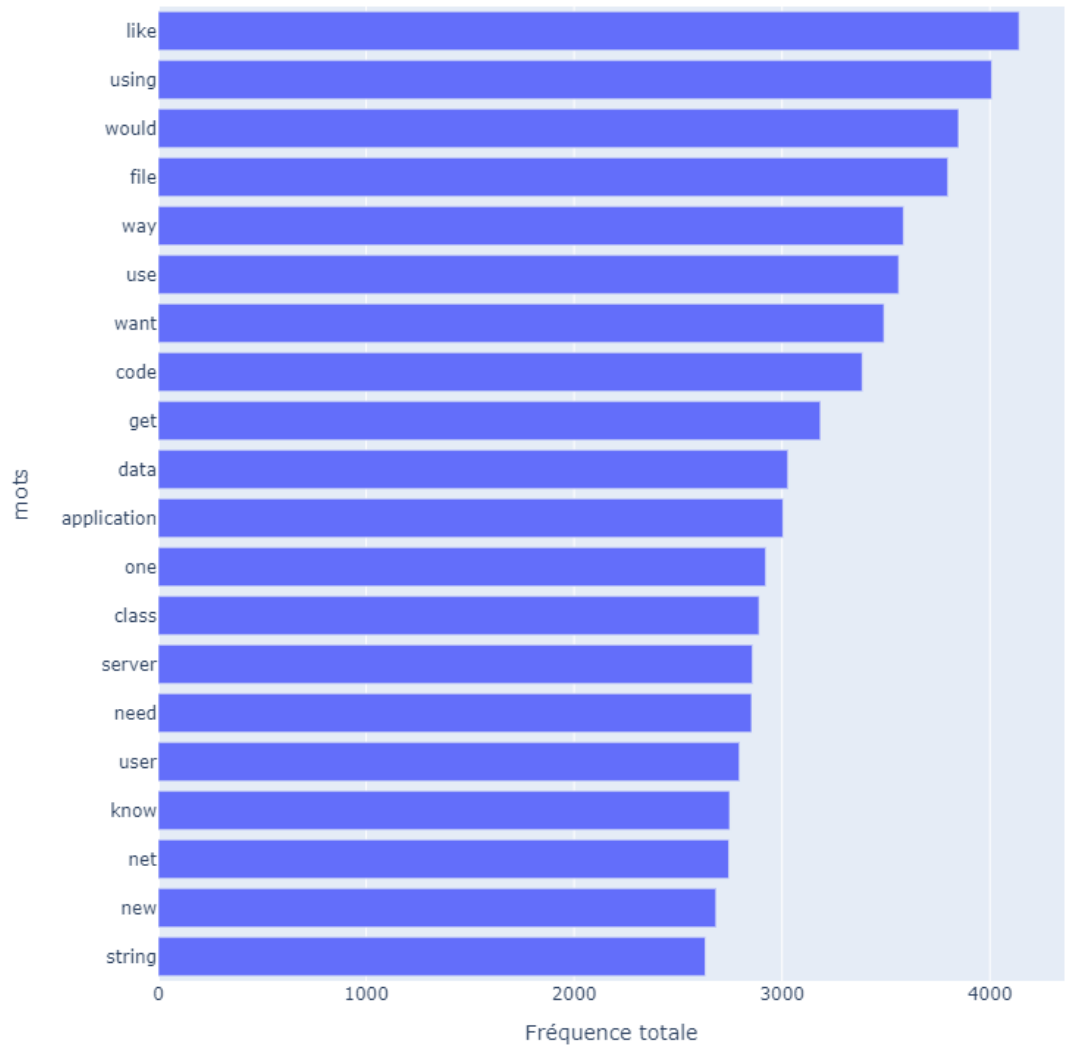
9. retirer les chevrons

Nous pouvons réaliser la représentation par un vecteur :

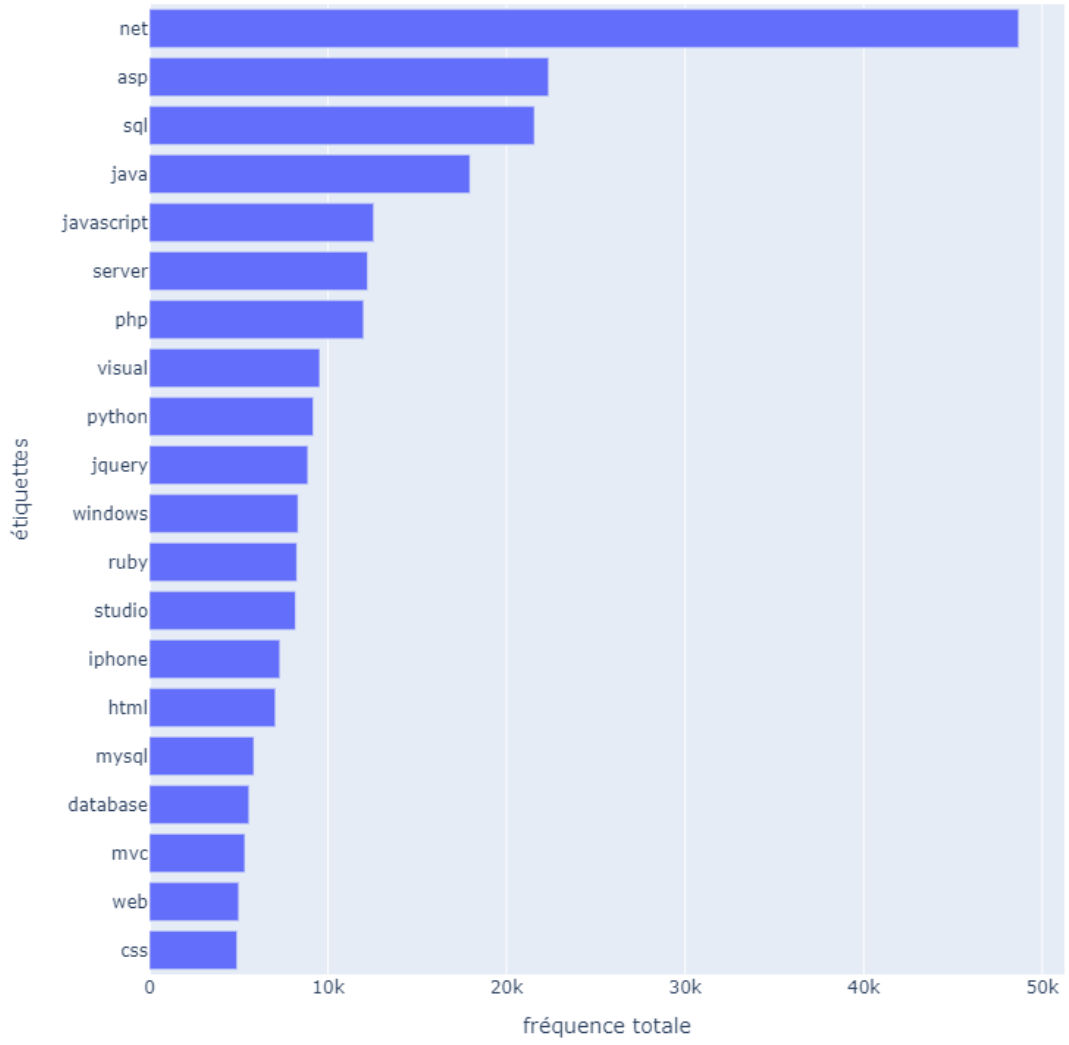
- Sac de mots (Bag-of-words model)
- TF-IDF (Term Frequency - Inverse Document Frequency)

Explorer

Analyse des 20 mots les plus fréquents



Analyse des 20 étiquettes les plus fréquentes

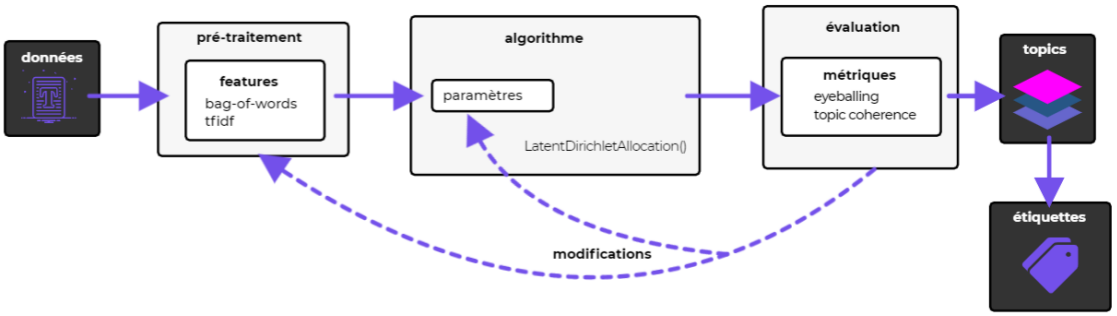


Modéliser

L''allocation de Dirichlet latente ou LDA est un modèle génératif probabiliste permettant d'expliquer des ensembles d'observations, par le moyen de groupes non observés, eux-mêmes définis par des similarités de données.

C'est une modélisation de sujets non supervisée que l'on peut représenter par ce processus :

Process de modélisation non supervisée *(dév)*



Latent Dirichlet Allocation

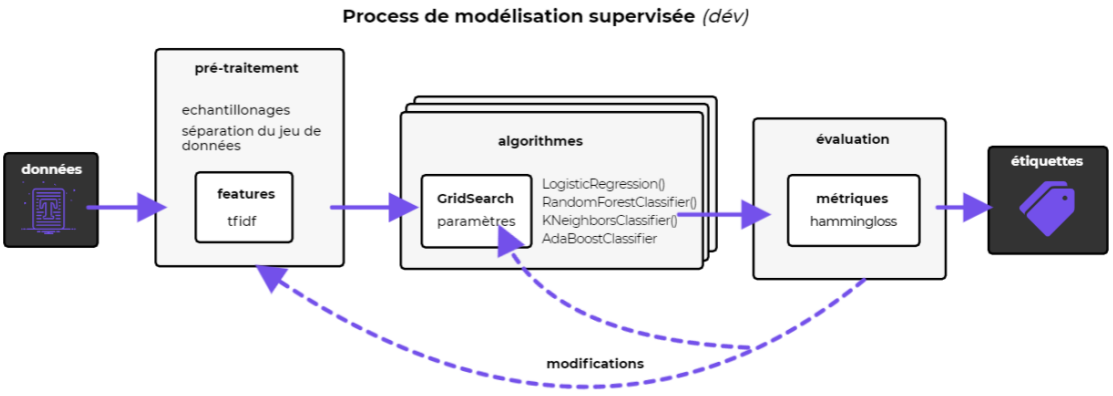
```
Entrée [8]: display_topics(lda, lda_posts_body_feature_names, 10)

-----------------------------------------
Topic 0:
user session login application password email page pdf web authentication
-----------------------------------------
-----------------------------------------
Topic 1:
database data sql table tables db entity server use way
-----------------------------------------
-----------------------------------------
Topic 2:
use good know one net code web looking question application
-----------------------------------------
-----------------------------------------
```

La LDA ne donne pas de résultat très probant et comme nous disposons d'étiquettes, il est plus judicieux de continuer vers une modélisation supervisée.

La Multilabel classification est la tâche qui consiste à assigner un ou plusieurs labels ayants pour valeur 0 ou 1. Dans notre cas, cela correspond à la présence de l'étiquette ou de son absence.

Certains estimateurs sont capables de générer d'office plusieurs étiquettes, d'autres ont besoin de méta-estimateur comme le MultiOutputClassifier.

Nous pouvons représenter le processus de modélisation supervisée :



Dans la classification multi-label, les prédictions sont des vecteurs d'étiquettes et, par conséquent, la prédiction peut être entièrement correcte, partiellement correcte ou totalement incorrecte. Les métriques par défaut accuracy, precision, recall ne captent pas cette notion.

Nous allons utiliser la métrique **Hamming Loss** qui est la fraction d'étiquettes qui sont incorrectement prédites sur l'ensemble de nos vecteurs. Elle prendra donc en compte les faux positifs et les faux négatifs. Elle est comprise entre 0 et 1, une HL = 0 indique l'absence totale d'erreur, cad plus sa valeur est petite plus la classification est performante.

On va utiliser également un DummyClassifier() qui effectue des prédictions à l'aide de règles simples.

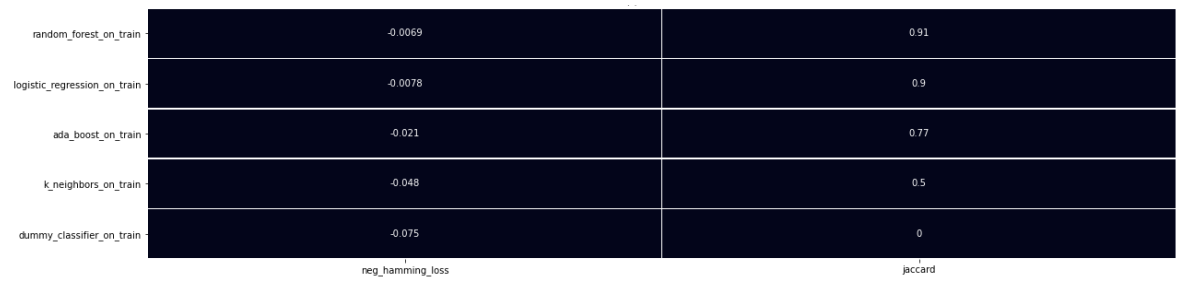
Ce classificateur est utile comme base de référence simple pour comparer avec d'autres classificateurs.

Le paramètre, strategy = “most_frequent”, prédit toujours l'étiquette la plus fréquente dans l'ensemble d'apprentissage.

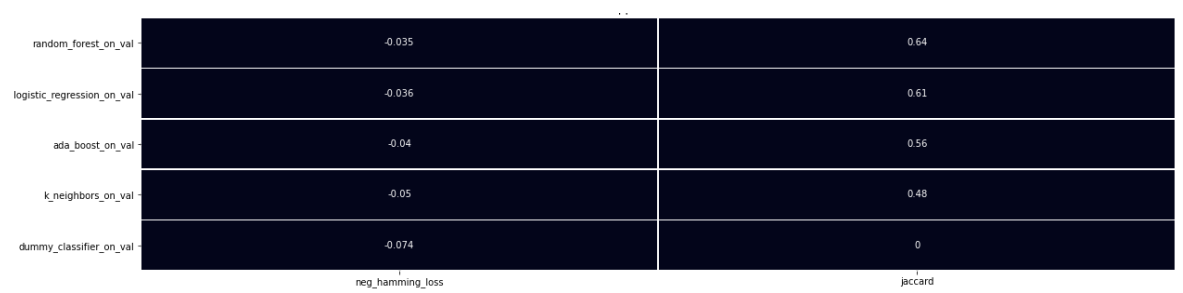
Dans notre cas, il va toujours prédire zéro 0 étant donné que l'on a une matrice creuse.

Nous pouvons ainsi comparer nos différents modèles après entraînement :

Rapport de classification sur **Train**



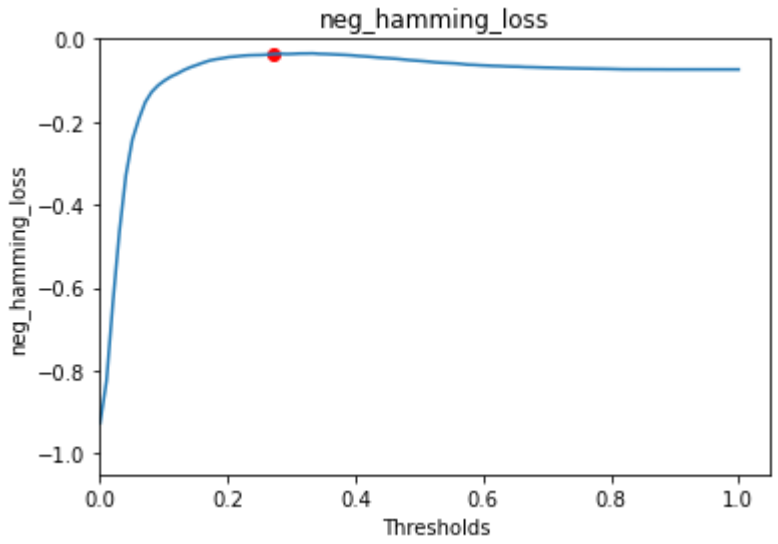
Rapport de classification sur **Val**

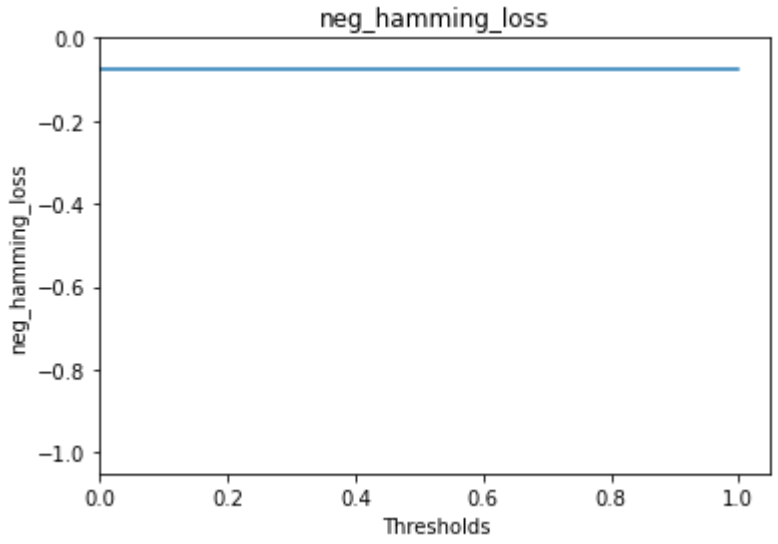


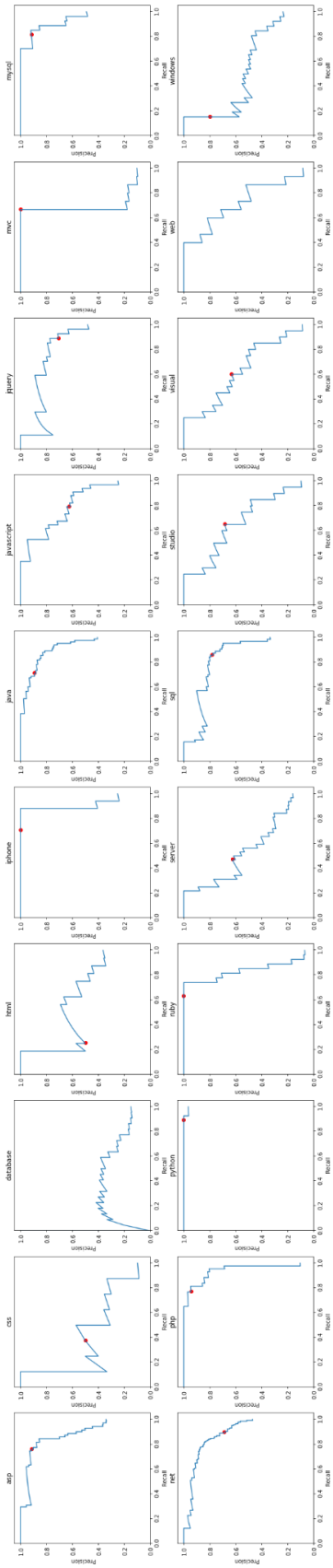
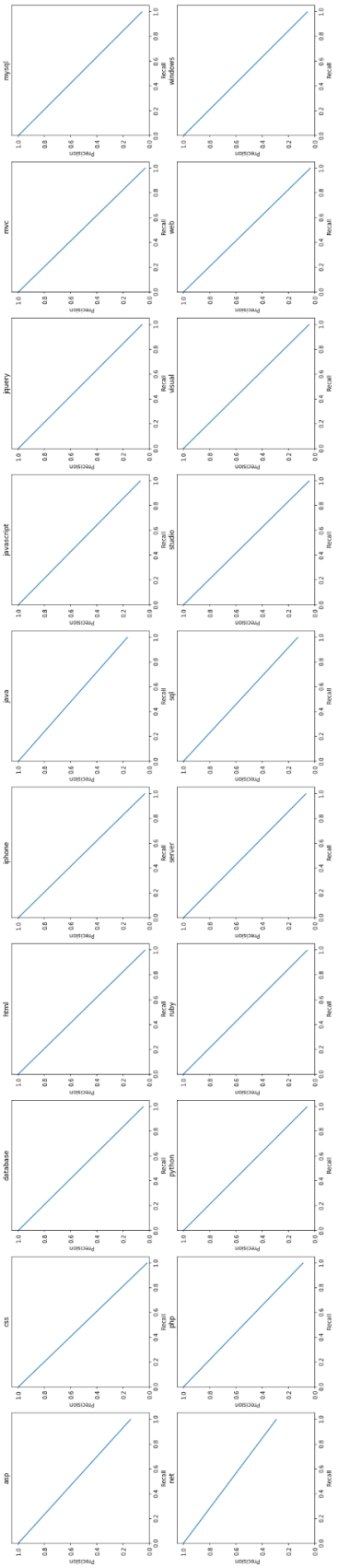
Le modèle ensembliste **Random forest** est le plus performant.

Comprendre

Le threshold est déterminé par la meilleure performance possible sur notre métrique **Hamming Loss**

randomforest


baseline


randomforest	baseline
	

On peut voir la performance de prédictions pour nos 20 étiquettes séparément.

Plus le point rouge est proche du coin haut droit, plus la prédiction est performante.

Sa position indique le trade-off entre faux positifs et faux négatifs.

Conclusion

Nous avons réussi à réaliser un modèle de prédictions assez performant pour être utilisé en production. Notre choix s'est porté sur un algorithme de classification supervisée, le random forest. On peut afficher une liste d'étiquettes de suggestions pour aider l'utilisateur à remplir le formulaire. Des améliorations sont évidemment possibles, on peut continuer à expérimenter, choisir un vocabulaire plus restreint, augmenter la taille des échantillons, proposer plus d'étiquettes, revoir l'implémentation du multiouput, appliquer un threshold différent par label, etc.