

Rapport

StackOverflow – Modélisation de prédiction des tags

Auteur	Alain ROUILLON		
Projet	Parcours OC Ingénieur Machine Learning		
Sujet	Catégorisation automatique des questions du site StackOverflow		
Révision	Date	Objet	Description
	02/05/2021	Initiale	
	03/05/2021	Review	Corrections Ajouts liens
	08/05/2021	Modification synthèse LDA	Ajout conclusions sur nouvelle modélisation

Table des matières

1. Description du projet.....	3
1.1 Contexte.....	3
1.4 Livrables.....	3
2. Préparation des données.....	5
2.1 Cleaning.....	5
2.2. Exploration des tags.....	5
2.3 Préparation du texte.....	6
2.4 Exploration du texte.....	7
2.5 Feature Engineering.....	7
4. Modélisations.....	8
4.1 Dataset.....	8
4.2 Modélisation non supervisée.....	8
4.3 Modélisation supervisée.....	10
5. Fonctionnement retenu.....	11
6. API.....	11
7. Axes d'amélioration.....	12

1. Description du projet

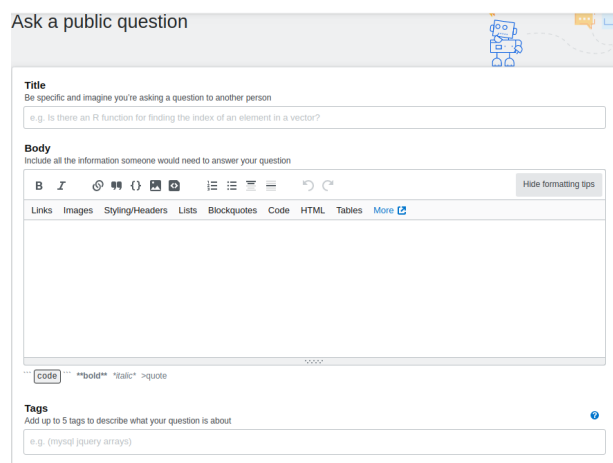
1.1 Contexte

Le site StackOverflow (<https://stackoverflow.com/>) offre la possibilité aux informaticiens de poster des questions à la communauté sur des problématiques rencontrées.

L'interface du site permet de saisir :

- un titre
- une description de la problématique rencontrée
- De coller du code pour préciser le problème rencontré
- D'attribuer des tags pour décrire le sujet

La saisie des tags, limitée à 5 choix parmi les tags paramétrés du site, permettra de répertorier la question et de driver le moteur de recherche en vue de recherche par les internautes.



1.2 Problématique

Le périmètre des problématiques rencontrées est extrêmement vaste, puisque le métier d'informaticien couvre des missions très diverses, allant du développement à l'ingénierie système, à l'administration de base de données, à la robotique, etc, ...

L'idée est donc de guider l'utilisateur en soumettant une proposition de tags pouvant correspondre à son post à partir des mots clés de celui-ci.

1.3 Démarche suivie

Pour ce projet nous avons retenu deux approches possibles :

- Une approche non supervisée
 - on cherche à dégager des topics des posts existants
 - la finalité est alors d'associer à ces topics les tags des posts pour lesquels ils sont prédominants
- Une approche supervisée
 - à partir des posts existants déjà taggués on cherche à modéliser le lien qui existe entre eux pour effectuer par la suite une probabilité de choix possibles
 - il s'agit d'une modélisation de classification multi-label

1.4 Livrables

L'objectif est de communiquer à l'utilisateur la suggestion des tags possibles au moment de la saisie par le portail du site. Pour cela on met en place une API qui peut être intégrée en appel REST par le site.

StackOverflow - Catégorisation automatique de questions

Pour ce projet l'API est interfacée pour permettre l'interaction avec les fonctions de prédictions soutenues par le(s) modèle(s) mis en place.

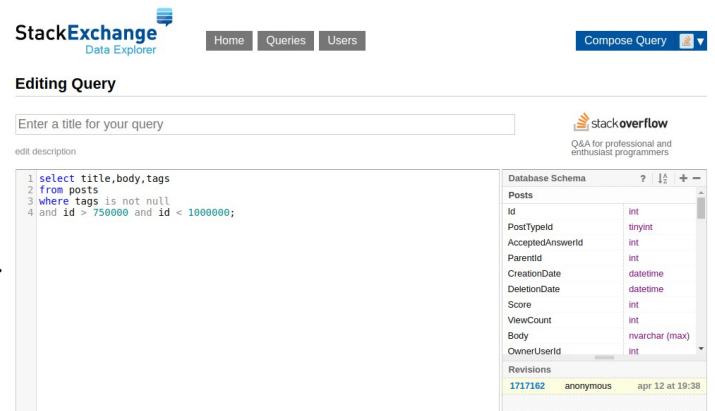
Project : <https://github.com/aldeponk/stack-overflow.git>

API : <http://localhost:5000/>

2. Préparation des données

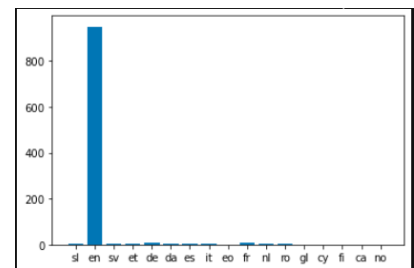
Les données sont récupérées depuis le site <https://data.stackexchange.com/stackoverflow>

Les données sont à récupérer par paquets de 50 000 enregistrements, Pour obtenir un dataset consistant l'idée retenue est de ne récupérer que les posts qui sont taggués.



2.1 Cleaning

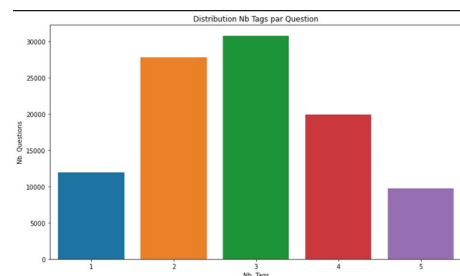
- Après analyse de la distribution des langues représentées on ne conserve que les posts rédigés en anglais
- Suppression des enregistrements dont le body du post est NaN
- Suppression des enregistrements dont les tags sont NaN



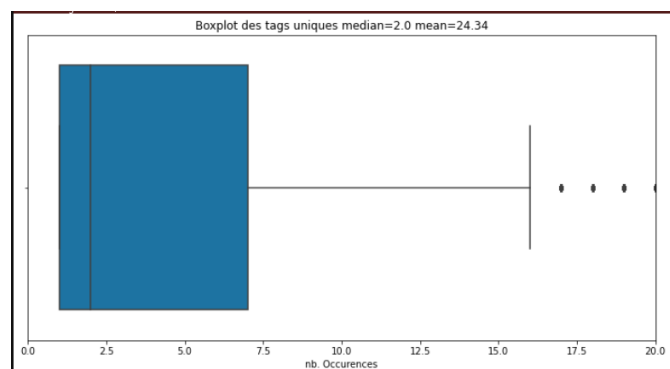
2.2. Exploration des tags

Sur le dataset étudié on recense 11812 tags distincts.

La distribution du nombre de tags par question présente un maximum de 3.



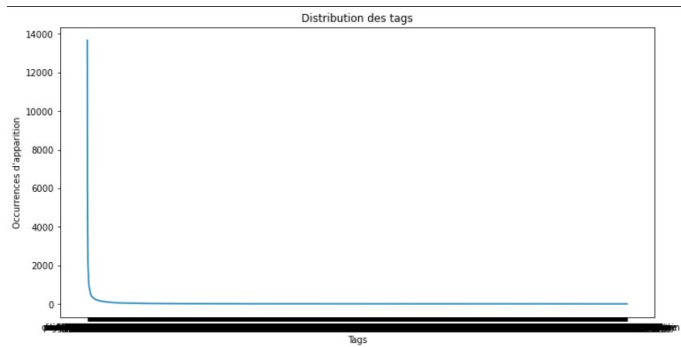
50 % des tags ne sont utilisés que 2 fois.



StackOverflow - Catégorisation automatique de questions

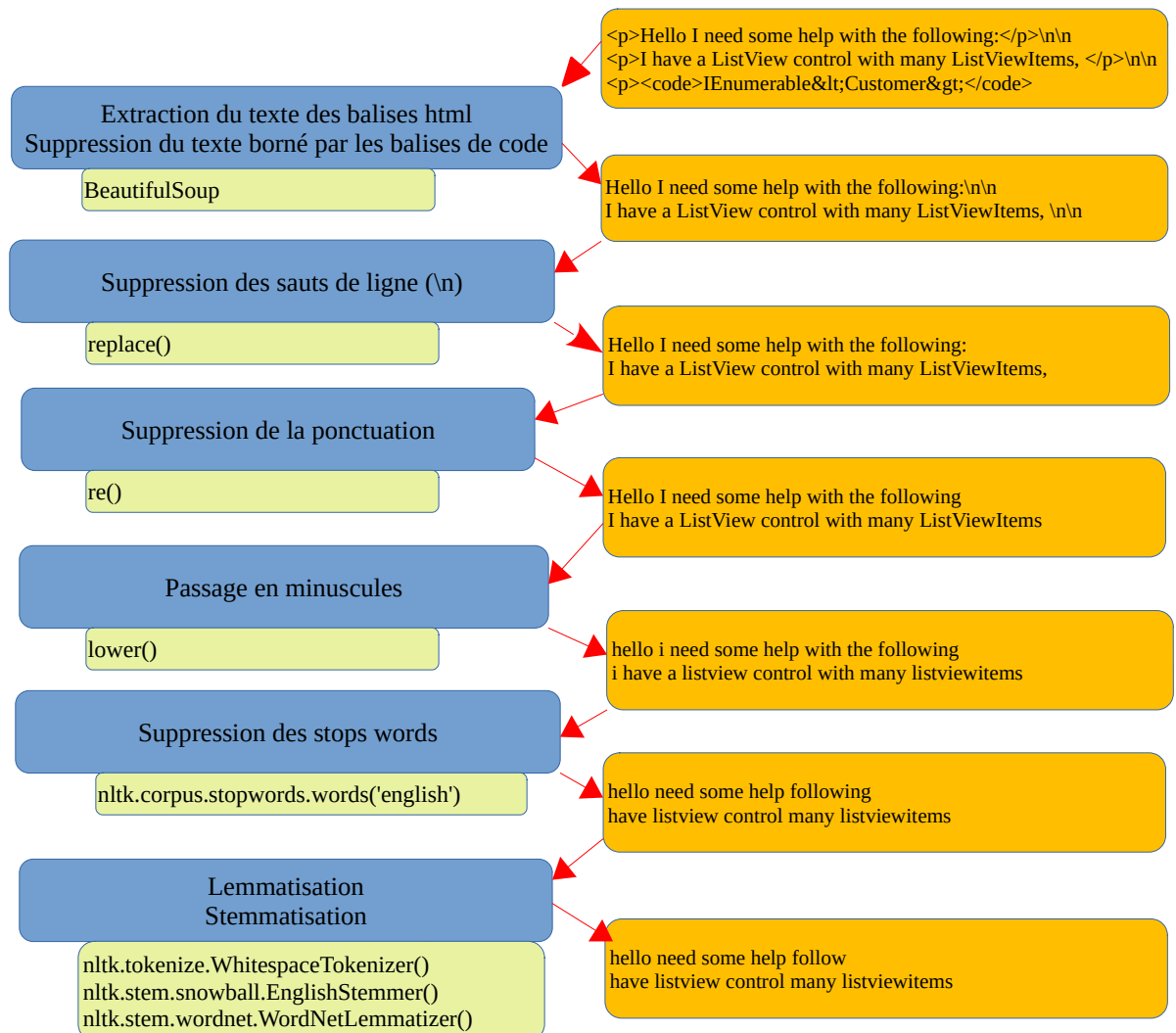
La distribution de la fréquence d'utilisation des tags montre que peu de tags sont souvent représentés, il sera donc très difficile d'entraîner un modèle performant pour les prédire.

Ce point est important à souligner car il va conditionner notre démarche de modélisation supervisée.



2.3 Préparation du texte

Afin d'effectuer notre modélisation nous effectuons un pré-processing des données textuelles portées par le corps des questions.



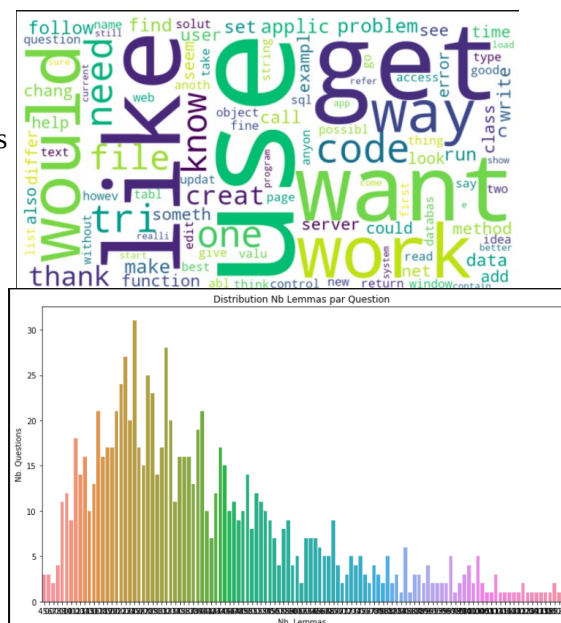
2.4 Exploration du texte

Après pré-traitement (voir point précédent) on fait un petit zoom sur le contenu des termes qui vont constituer le corpus.

On constate visuellement que les termes servant à introduire les questions restent très présents, ce qui ne sert pas directement notre objectif. Cela pourra constituer une piste d'amélioration (voir point 7)

La distribution de termes racine par question est presque normale avec un léger décalage à gauche.

Globalement les bodies des posts qui vont servir au training sont bien distribués en termes d'apport de mots racine.



2.5 Feature Engineering

En vue de la modélisation on procède préalablement à une étape de feature engineering portant sur la colonne du corps du post (nommé document dans la suite). L'idée est de vectoriser le document pour apporter de l'information à l'algorithme d'apprentissage :

- indicateur de fréquence de chaque terme dans le document
- notion de séquençement du terme (ordre)

Plusieurs techniques ont été testées :

- Bag of Words
- Count Vectorizer
- TF-IDF

TF-IDF est la méthode qui permet de mettre en place le plus d'informations, en donnant un indicateur calculé comme étant le produit de la fréquence du terme dans un document (TF) et de la fréquence de document dans lequel le terme apparaît (IDF) :

$$TF-IDF = TF(t,d) * IDF(t)$$

Cela permet de coupler à un terme donné un facteur de fréquence à une dimension tenant compte de 2 dimensions.

4. Modélisations

4.1 Dataset

Par la suite, pour des problématiques de ressources machine, le dataset utilisé en entrée de training est limité à 9 342 enregistrements.

4.2 Modélisation non supervisée

L'objectif est de construire un moyen de matérialiser le dataset dont le contenu est inconnu a priori.

L'algorithme Latent Dirichlet Allocation (LDA) permet de construire des topics représentant les thèmes abordés dans les documents en entrée.

L'algorithme repose sur l'identification d'un nombre fini de topics donné en entrée, et cherche pour chaque terme de chaque document à maximiser la probabilité que le document soit rattaché au thème t construit à partir de l'ajout du nouveau terme, et à maximiser la probabilité que le thème t soit assigné au terme.

Nous avons choisi l'implémentation LDAMulticore du package gensim.

L'entraînement du modèle est passé par la mise en place d'une baseline avec les paramètres par défaut, puis par une phase de détermination des meilleurs paramètres par itération sur des plages de valeurs candidates pour les paramètres principaux :

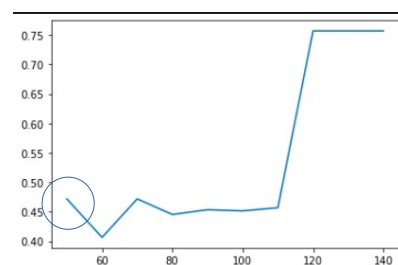
Paramètre	Description
Topics	Nombre de thèmes à construire
alpha	Contrainte de l'algorithme à considérer qu'un document est composé de plus ou moins de topics. Plus alpha est bas, plus l'algorithme tendra à associer peu de topics dominants.
beta	Contrainte de l'algorithme à considérer qu'un topic est composé de plus ou moins de termes. Plus beta est bas, plus l'algorithme va considérer que les topics sont composés d'un nombre de termes limité.

- Valeur optimale pour le nombre de topics

On retient la valeur la plus haute précédant la première baisse du score de cohérence. Ici on retient donc la valeur 50

- Alpha, beta

L'optimisation de ces deux paramètres a été effectuée en cherchant à maximiser la variance de distribution des topics dominants. Le tableau ci-après montre deux trainings du modèle avec une mise en évidence de la variance obtenue.



StackOverflow - Catégorisation automatique de questions

Variance de distribution faible à nulle	Variance de distribution élevée																						
Alpha, beta pour le score de cohérence max	Alpha, beta pour un score de cohérence dans la moyenne, mais supérieur à la valeur de baseline																						
<table><tr><th>Validation_Set</th><th>Topics</th><th>Alpha</th><th>Beta</th><th>Coherence</th></tr><tr><td>308</td><td>100% Corpus</td><td>50</td><td>0.31</td><td>0.9099999999999999</td><td>0.672459</td></tr></table>	Validation_Set	Topics	Alpha	Beta	Coherence	308	100% Corpus	50	0.31	0.9099999999999999	0.672459	<table><tr><th>Validation_Set</th><th>Topics</th><th>Alpha</th><th>Beta</th><th>Coherence</th></tr><tr><td>300</td><td>100% Corpus</td><td>50</td><td>0.01</td><td>0.01</td><td>0.471724</td></tr></table>	Validation_Set	Topics	Alpha	Beta	Coherence	300	100% Corpus	50	0.01	0.01	0.471724
Validation_Set	Topics	Alpha	Beta	Coherence																			
308	100% Corpus	50	0.31	0.9099999999999999	0.672459																		
Validation_Set	Topics	Alpha	Beta	Coherence																			
300	100% Corpus	50	0.01	0.01	0.471724																		
Distributions obtenues	Distributions obtenues																						
Visualisation éclatée	Visualisation éclatée																						
On constate que chaque topic a un poids de 2 % ce qui ne permettrait pas de faire de prédiction, chaque topic ayant la même probabilité de matcher un document	On constate que les topics sont bien mieux caractérisés, avec des poids différents, ce qui permettra d’effectuer des prédictions bien discriminantes																						

Afin d'effectuer une suggestion de tags on construit une matrice à partir du topic dominant de chaque post, dont on connaît les tags.

En traitant récursivement le dataset on somme les poids des topics qu'on affecte aux tags des documents.

On obtient ainsi une matrice (tags, Topics) avec pour chaque tag le pourcentage sommé pour chaque topic.

Pour uniformiser les données on normalise

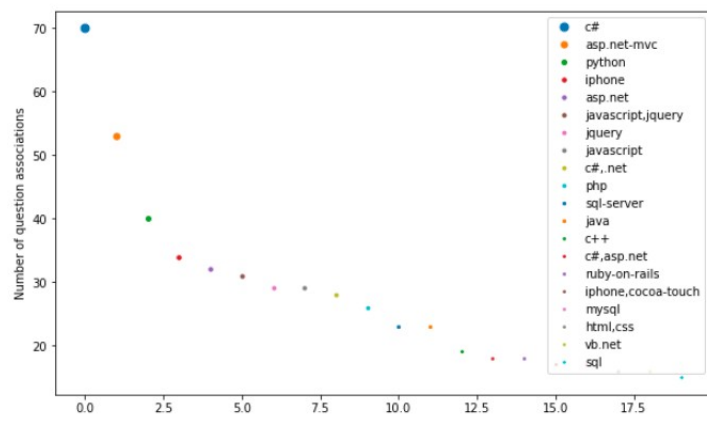
la matrice obtenue.

Cette matrice donne alors les n tags associés à un topic qui ressort de la prédiction pour un nouveau document.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
.emf	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.000000	0.000000	0.0	0.000000
.htaccess	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.243982	0.000000	0.0	0.000000	0.000000	0.0	0.000000
.net	0.0	0.692468	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.349068	0.829108	0.0	0.480571	0.557755	0.0	0.986039
.net-2.0	0.0	0.016451	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.000000	0.000000	0.0	0.000000
.net-3.5	0.0	0.046456	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.121358	0.0	0.000000	0.000000	0.0	0.000000

4.3 Modélisation supervisée

Sur le périmètre limité aux 9 342 enregistrements la distribution de tags est la suivante :



Nous partons sur une modélisation pour les 20 tags les plus fréquents. Il s'agit d'une classification multi-label,

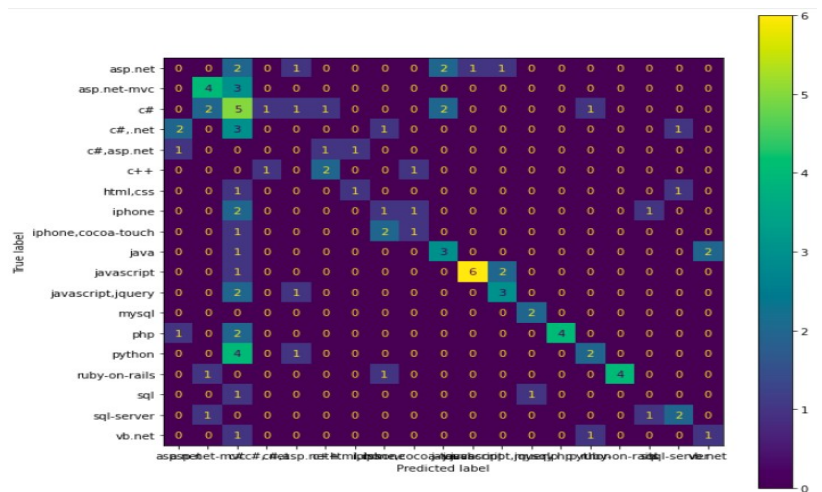
Le modèle choisi est XGBoostClassifier.

Le réglage des hyperparamètres est fait par pipeline à partir de GridSearchCV.

L'estimateur des performances du modèle est l'accuracy.

L'accuracy obtenue sur le jeu de test est de 0,39.

La matrice de confusion montre les cas d'erreurs rencontrés.



Ce résultat n'est pas très bon, il ne permet pas d'envisager une mise en production du modèle en l'état.

L'amélioration des résultats est possible en intégrant un nombre de cas plus important pour couvrir plus de combinaisons. Ce point nécessite un plus gros dataset, avec donc plus de ressources machine pour processor.

5. Fonctionnement retenu


Pour l'API nous retenons de fonctionner en couplant les deux modèles :

- Modèle supervisé : accuracy = 0,39
- Modèle non supervisé : coherence score = 0,47

L'API ressort les suggestions de tags effectuée par les deux modèles. Ici nous laissons ces deux listes en l'état pour permettre d'illustrer les résultats obtenus par les deux méthodes. Dans une application production il faudrait merger ces deux listes.

6. API

L'API est implémentée par Flask, le point d'entrée est servi par un serveur local (pas de déploiement internet sur un dsn public).

Splash screen	
 <p>Welcome to StackOverflow tagging API !</p> <p>Please have a GO !</p>	
Invite de saisie de la question et bouton de soumission	Résultats de la prédiction
<p>Question Stackoverflow:</p> <p>Hello, I am developing a new application in django with html and javascript and I need to create a new table in a sql-server database. Can you tell me how to do it ? Thanks !</p> <p>SEND</p>	<p>Question Stackoverflow:</p> <p>Hello, I am developing a new application in django with html and javascript and I need to create a new table in a sql-server database. Can you tell me how to do it ? Thanks !</p> <p>Tags suggérés (supervisé)</p> <p>sql sql-server c#</p> <p>Tags suggérés (non supervisé)</p> <p>design-patterns django python django-urls django-views</p> <p>←</p>

7. Axes d'amélioration

Général :

- Enrichir les stopwords pour éliminer les termes inutiles pour l'objectif : mots question, termes d'interaction de dialogue, etc, ...
- Augmenter taille du dataset
- GPU/RAM

Non supervisé

- Dataset plus large : en augmentant le nombre de données les calculs sont plus longs, mais surtout les combinaisons plus importantes, et donc le calcul de probabilité a de meilleures chances de donner des résultats pertinents.

Supervisé

- CNN : un réseau de neurone pourrait améliorer l'accuracy en mettant en place plusieurs couches cachées qui serviraient à affiner la classification, et donc réduire les cas de faux positifs.