Janvier 2019

StackOverflow

Projet OpenClassrooms

Par Xavier Montamat

Problématique

Catégoriser automatiquement des questions techniques sur StackOverflow.

Chaque question étant déjà actuellement catégorisée par l'utilisateur.

Il nous faudra choisir entre une méthode supervisé ou non pour répondre au problème.

Axes d’approche

Formatage du texte

- Récupérer un maximum de texte
- Réduire les features grâce aux techniques apprisent
- Transformer nos posts en matrice de mots

La prédiction

- Classification multi classes nécessaire
- Optimisation pour réduire le temps d'entraînement
- Analyser les erreurs
- Effectuer une classification non supervisée

Plan de réalisation

Pré traitement données texte

- Récupération des données
- De texte à mots
- Stemming
- Fréquences minimum
- Association de tags
- Réduction des features

Modèlisation

- Préparation, multilabels
- Algorithme supervisé
- Algorithme non supervisé

Modèle final

- Comparaison des résultats
- Conclusion
- Axes d’amélioration

Pré traitement

Le prétraitement des données est la majeure partie de ce projet.

Il nous faut d’abord transformer nos données texte sous une forme exploitable.

Puis il faudra réduire le nombre de feature et de targets avant de pouvoir passer à la modelisation

Récupération des données

Base SQL complète

Trop de données

Selection pertinentes

- Champs tous présents
- Texte minimum
- Question répondue

Corpus final

43K posts

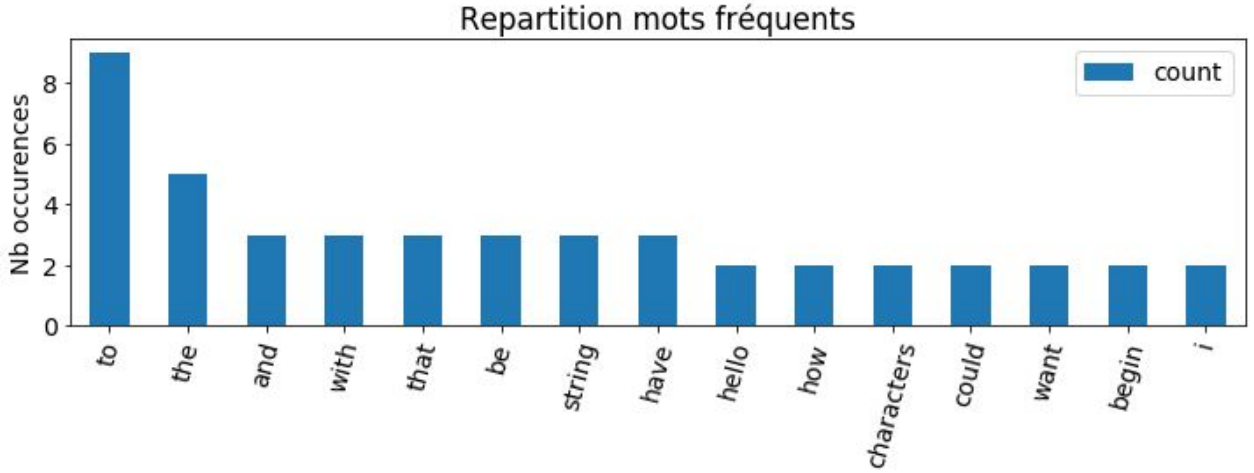
Transformation de text

Balises HTML - BeautifulSoup

Minuscules

Caractères spéciaux

Stop words - NLTK



Stemming / Lemming

Racine de mots - Stemmer

- Retrait préfixes / suffixes
- Permet de retrouver la racine
- Association sens commun
- Lancaster / Porter / Snowball

Lemmatizer

- Associe des mots dérivés à leur parent
- Ne dénature pas l'orthographe

Stemmer exemples

- Swapped == Swap
- Functions == Funct

Lemmatizer exemples

- Swapped != Swap
- Functions == Function

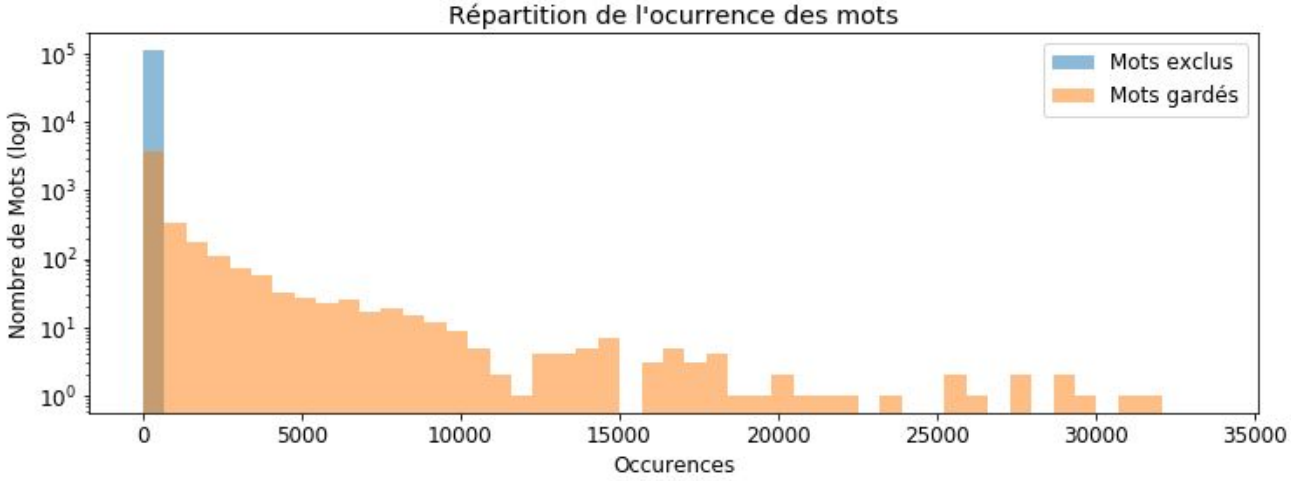
Fréquences de mots

Corpus

- 4.6 Millions de mots
- 115K uniques
- 80% utilisés moins de 5 fois

Filtre fréquence minimum

- 1 fois sur 1000 posts = 43
- Élimine 95.8% du vocabulaire
- En gardant 91.2 % du corpus total



Vocabulaire restant

4708

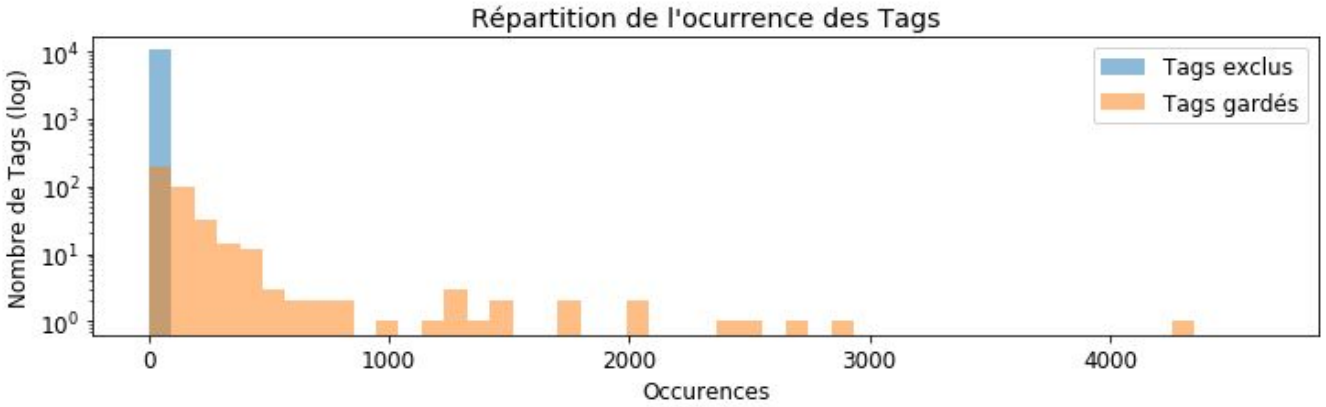
Fréquences des tags

Catégories

- 133K tags renseignés
- 11K catégories
- 80% utilisés moins de 6 fois

Filtre fréquence minimum

- 1 fois sur 1000 posts = 43
- Élimine 96.5% des catégories
- En gardant 68.3 % des tags totaux



Catégories restantes

379

Réduction de dimension cible

Catégories (données cibles)

- 379 restantes
- Similarités probables
- But : moins de 100 cibles

Difficultés

- Pas de suppression
- Garder un label
- Favoriser les tags fréquents

Exemples

'asp.net',
'asp.net-core',
'asp.net-mvc',
'asp.net-mvc-3',
'asp.net-mvc-4',
'asp.net-web-api',

'visual-studio',
'visual-studio-2008',
'visual-studio-2010',
'visual-studio-2012',
'visual-studio-2013',
'visual-studio-2015',

Réduction de dimension cible

Solution

- Solution manuelle
- Matrice de corrélation
- Score corrélation adapté à la fréquence
- Associations parent > enfants

Résultats

- 284 enfants > 72 parents
- +23 orphelins = **95 catégories**

Exemples associations

	replace_with
visual-studio	NaN
visual-studio-2008	visual-studio
visual-studio-2010	NaN
visual-studio-2012	visual-studio
visual-studio-2013	visual-studio
visual-studio-2015	visual-studio

	replace_with
asp.net	NaN
asp.net-core	visual-studio
asp.net-mvc	NaN
asp.net-mvc-3	asp.net-mvc
asp.net-mvc-4	asp.net-mvc
asp.net-web-api	asp.net-mvc

Réduction de features

Mots de vocabulaire

- 4708 restants
- Beaucoup de mots courants
- Ne garder que les mots ‘utiles’

Difficultés

- Evaluer ‘utilité’ sur toutes categories
- Limiter perte d’information

Exemples de mots top 100 (fréquence) peu pertinents

'valu', 'cod', 'id', 'new', 'get', 'work', ‘want’, ‘would’, ‘problem’, ’exampl’ ...

Réduction de features

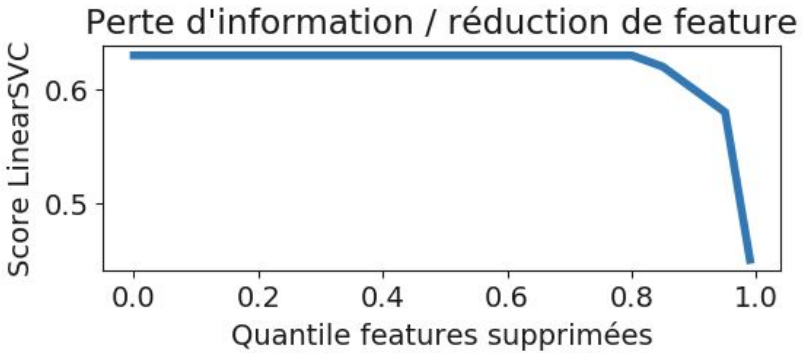
Solution

- Score basé sur coeff LinearSVC
- Précalcul performance avec 100% des features
- Evaluation de la réduction sur les prédictions

Resultats

- Choix 0% perte de précision
- 80% de features écartées
- 942 features restantes

	NB tags influencés	Score moyen
python	14	5.683
php	7	5.102
android	12	4.977
c#	11	4.783
cout	1	4.700
c++	4	4.580



Modèles de classification

Maintenant que nos features et cibles sont correctement formatées, nous pouvons appliquer nos modèles de classification.

Nous allons tout d'abord utiliser une méthode supervisée.

Puis une non supervisée.

Et enfin, comparer les deux

Préparation

Transformations

- TF-IDF pour nos features
- Poids de 2 pour les titres, 1 pour le corps
- Tags en targets

Train / Test split

- 75/25

Méthode supervisée

Problem_transformation

- Binary Relevance
- Classifier Chain

Modèles testés

- Random Forests
- Naive Bayes, Gaussian & Multinomial
- KNN
- LDA
- Linear SVC

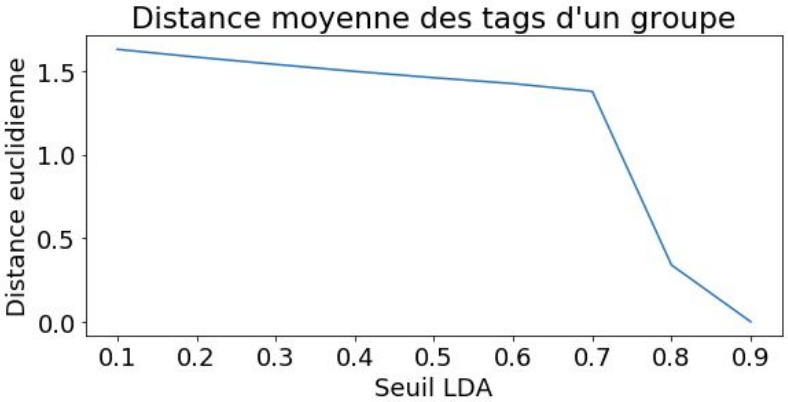
Modèle non supervisé

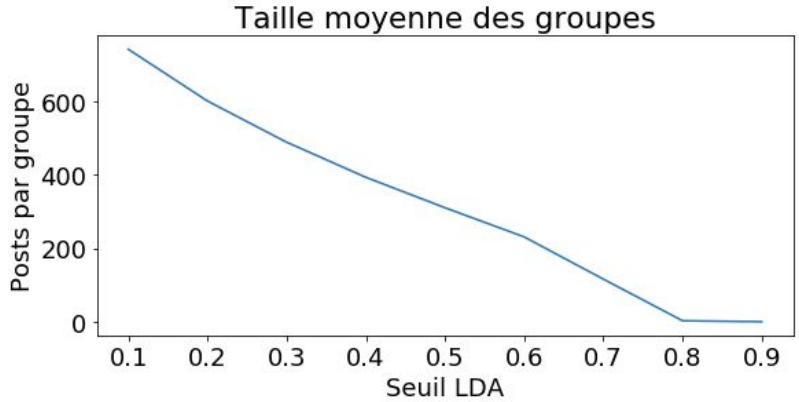
LDA - Latent Dirichlet Allocation

- N composants = 95
- Se rapprocher des tags

Variation du seuil

- Précision vs Taille de groupes
- Seuil de 0.25 pour une taille de groupe moyenne de 541





Modèle final

Nous comparerons les résultats des deux algorithmes

Rappelons les avantages et inconvénients de chacun

Exposition des améliorations possibles

Résultats modèle supervisé

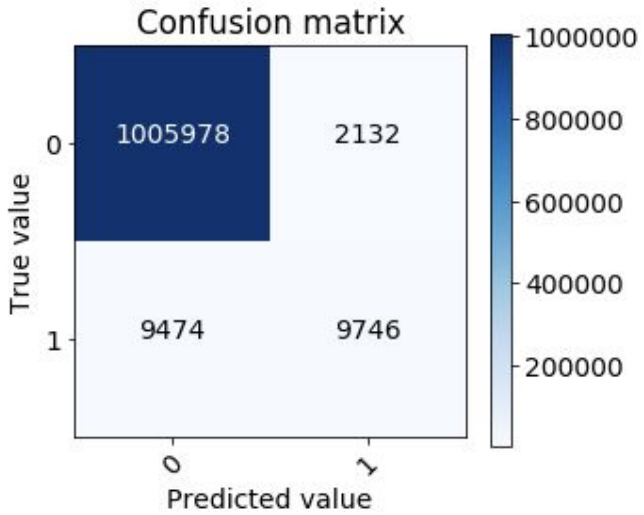
LinearSVC

- C=1, penalty l2
- Binary Relevance

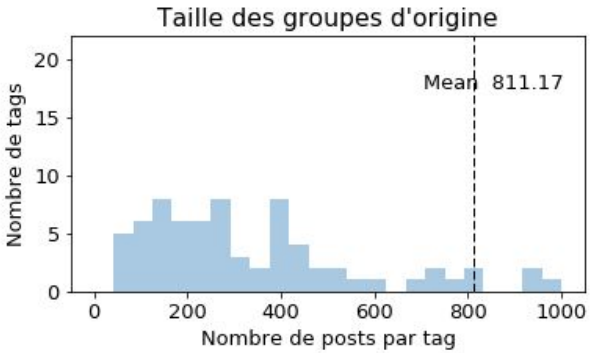
Scores

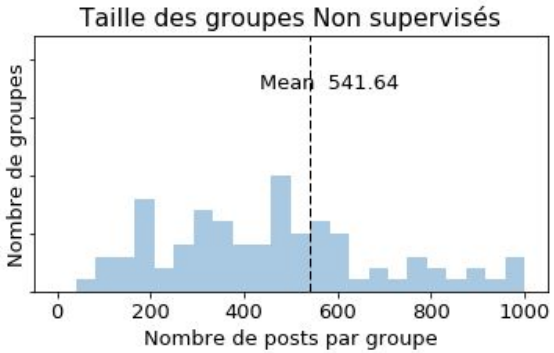
- Précision : 0.82
- Recall : 0.51
- F1-score : 0.63
- Jaccard Similarity : 0.53

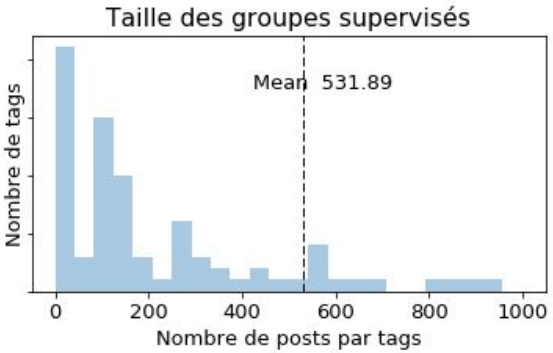
531 posts par tag en moyenne vs 811

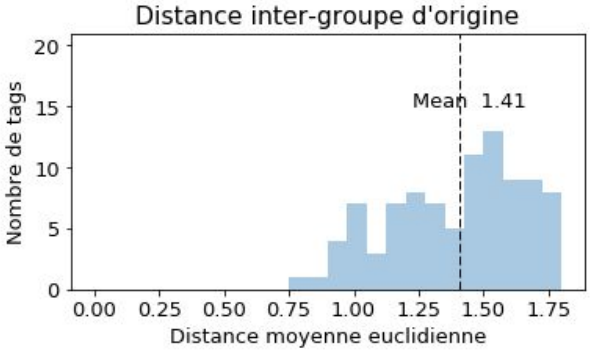


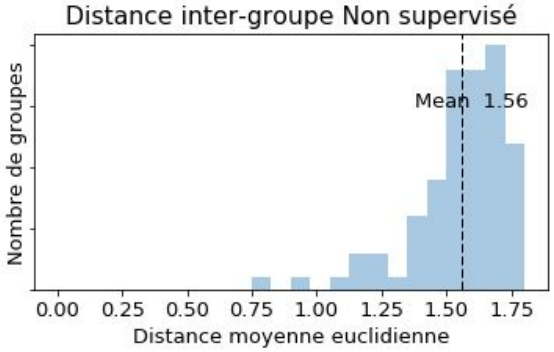
Comparaison des résultats

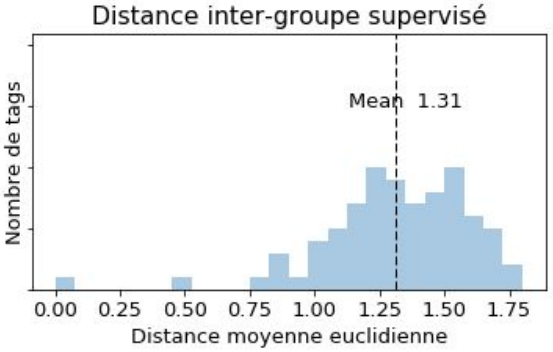












API

Actions effectuées

- Tags HTML, lowercase, ponctuations
- Stop words supprimés
- Lancaster Stemmer
- Comparaison entre mots transformé et features
- Prédiction des tags par modèle LinearSVC pré entraîné

Conclusion

Notre algorithme final supervisé se trompe peu et est donc utile dans un cadre de prédiction partielle de tags.

Il peut être utilisé pour tagger des posts passé mal classés. Ou bien assister un utilisateur.

Améliorations possibles

Suggestion d’un top 10 tags pertinents
Pourrait compléter notre modèle
Ajouter l’historique utilisateur

Questions

?

Liens

Lien GitHub du projet

https://github.com/xmontamat/OCR_Project6_StackOverFlow

Lien de l'API en ligne

http://xmontamat.pythonanywhere.com/