OPENCLASSROOMS

Catégorisation automatique des questions Stackoverflow

Projet 6

Parcours Data Scientist

/02/2020

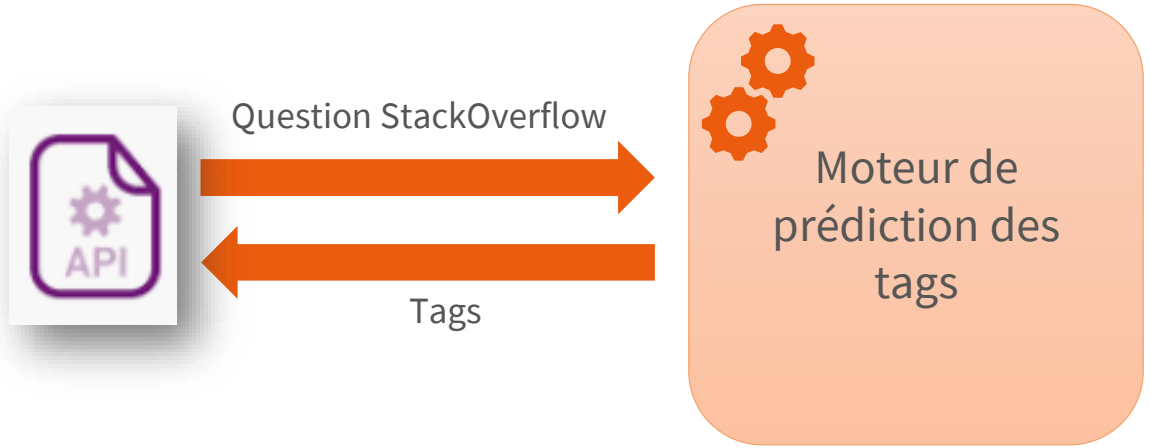
AYOUB MCHAREK

SOMMAIRE

1. PRÉSENTATION DE LA PROBLÉMATIQUE

Enoncé :

- A l'aide des méthodes **se traitement de langage naturel**
- Il faut élaborer une **API**
- Pour suggérer des **tags pertinents**
- à partir de l'historique des questions sur **StackExchange**



Algorithmes de traitement de langage naturel :

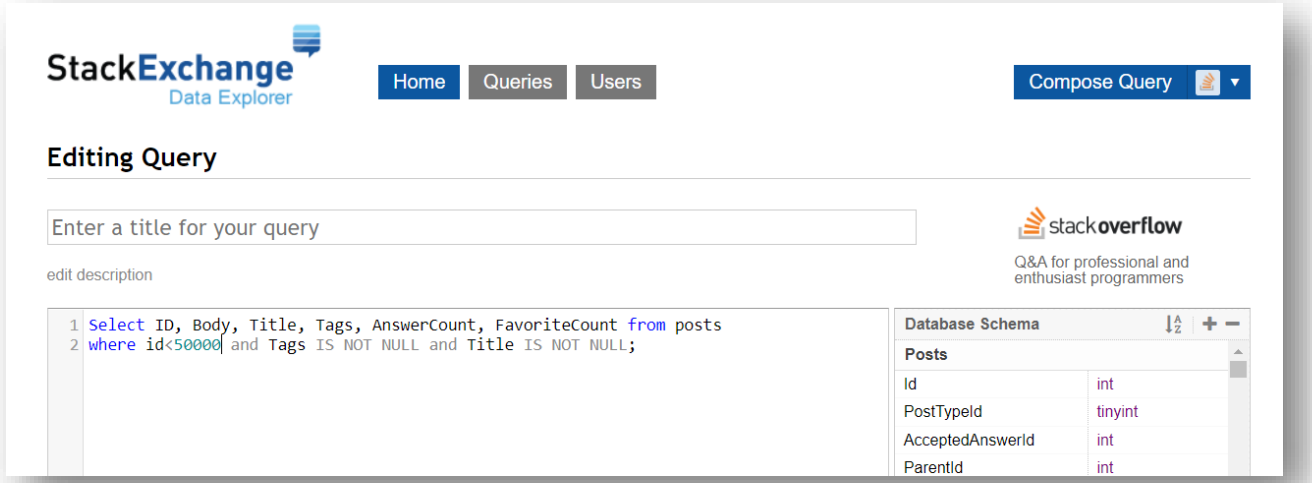
- **Non supervisée :** détecter les sujets latents abordés dans un corpus de documents, ensuite assigner les sujets détectés à ces différents questions.
- **Supervisée :** la classification multi-labels en utilisant des tokens extraites du corpus comme variables en entrée et les tags dont on dispose comme variables cibles.

Contraintes :

- **Prétraitement :** Des données non structurées pour obtenir un jeu de données exploitable
- **Techniques de réduction des dimensions**
- **Mettre en place une méthode d'évaluation propre**
- **Utiliser un logiciel de gestion de versions, Git**

2. ANALYSE EXPLORATION ET FEATURE ENGINEERING

Données Sources	Contenu	Variables - 5	
		3 variables Qualitatives nominales	2 variables qualitatives discrètes
BDD publique 3 extractions fichiers csv	103.676 lignes des questions	Body : question en format html Title : titre de la demande Tags : les résultats précedentes	AnswerCount : le nombre des réponses à la question FavoritCount : le nombre indiquant la popularité de la question

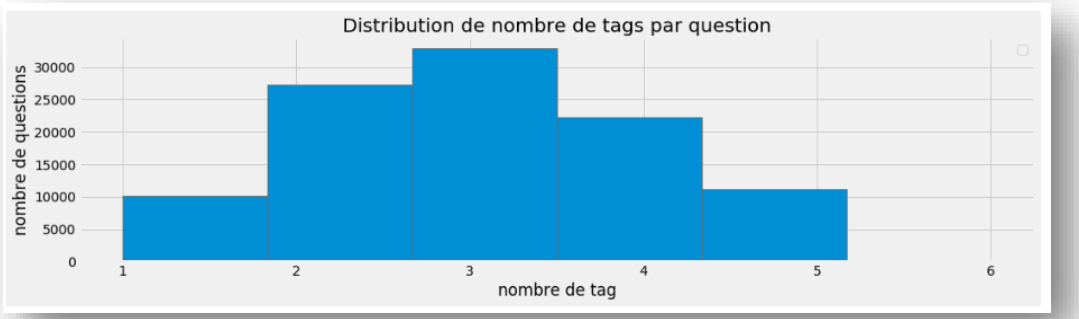


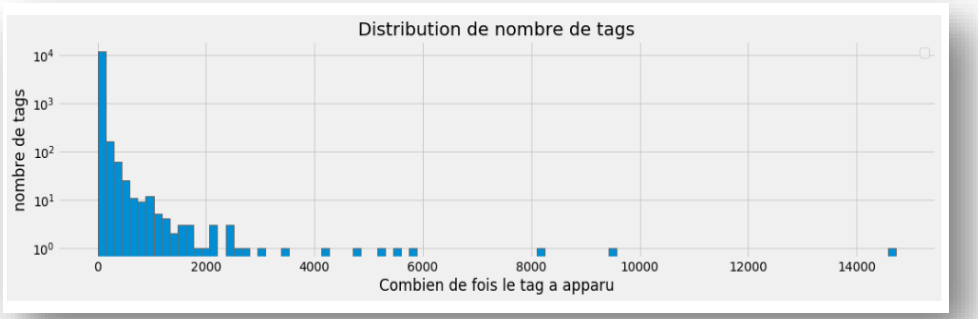
But de l'analyse et du feature engineering :

- Des visuels pour comprendre l'état des données
- Nettoyage et préparation des données textuelles
- Transformations des données brutes en données exploitables
  - Tokenization
  - StopWords
  - Stemming et Lemmatisation
  - Bag Of Words et TF-IDF

2. ANALYSE EXPLORATION ET FEATURE ENGINEERING

- L'analyse m'a permis de mieux comprendre le comportement de mes variables, mais j'ai choisi de ne pas prendre en considérations ces résultats pour faire un nettoyage






```
'c#': 14725,
'.net': 9511,
'java': 8160,
'c++': 5759,
'asp.net': 5526,
'javascript': 5165,
'python': 4742,
'php': 4212,
'sql': 3388,
'sql-server': 3036,
'jquery': 2716,
'iphone': 2552,
'html': 2497,
'c': 2406,
'windows': 2403,
'asp.net-mvc': 2192,
'wpf': 2137,
'mysql': 2123,
'database': 1933,
```

Traitement des données :

- **BeautifulSoup** : Préparation Body : HTML → TEXT
- **NLTK. RegexpTokenizer(r'\S+') :** le découpage des questions en « mots »
- **NLTK. Stop words, lemming and stemming :** suppression des mots courants, revenir à la forme canonique et racinisation
- **Gensim. Bigrammes & Trigrammes :** «sql_server», «internet_explor_7» : paramèters : **(min_count=5, threshold=50**)
- **Bag-Of-Words :** la représentation de chaque question par un vecteur de la taille du vocabulaire avec le nombre d'apparition du mot comme valeur pour chaque variable
- **TF_IDF : term frequency – inverse document frequency:** augmente proportionnellement au nombre de fois qu'un mot apparaît dans le document, ensuite compensée par le nombre de documents dans le corpus qui contiennent le mot
    - Plus de néttoyage : **(max_df = 0.5, min_df = 10)**

2. ANALYSE EXPLORATION ET FEATURE ENGINEERING

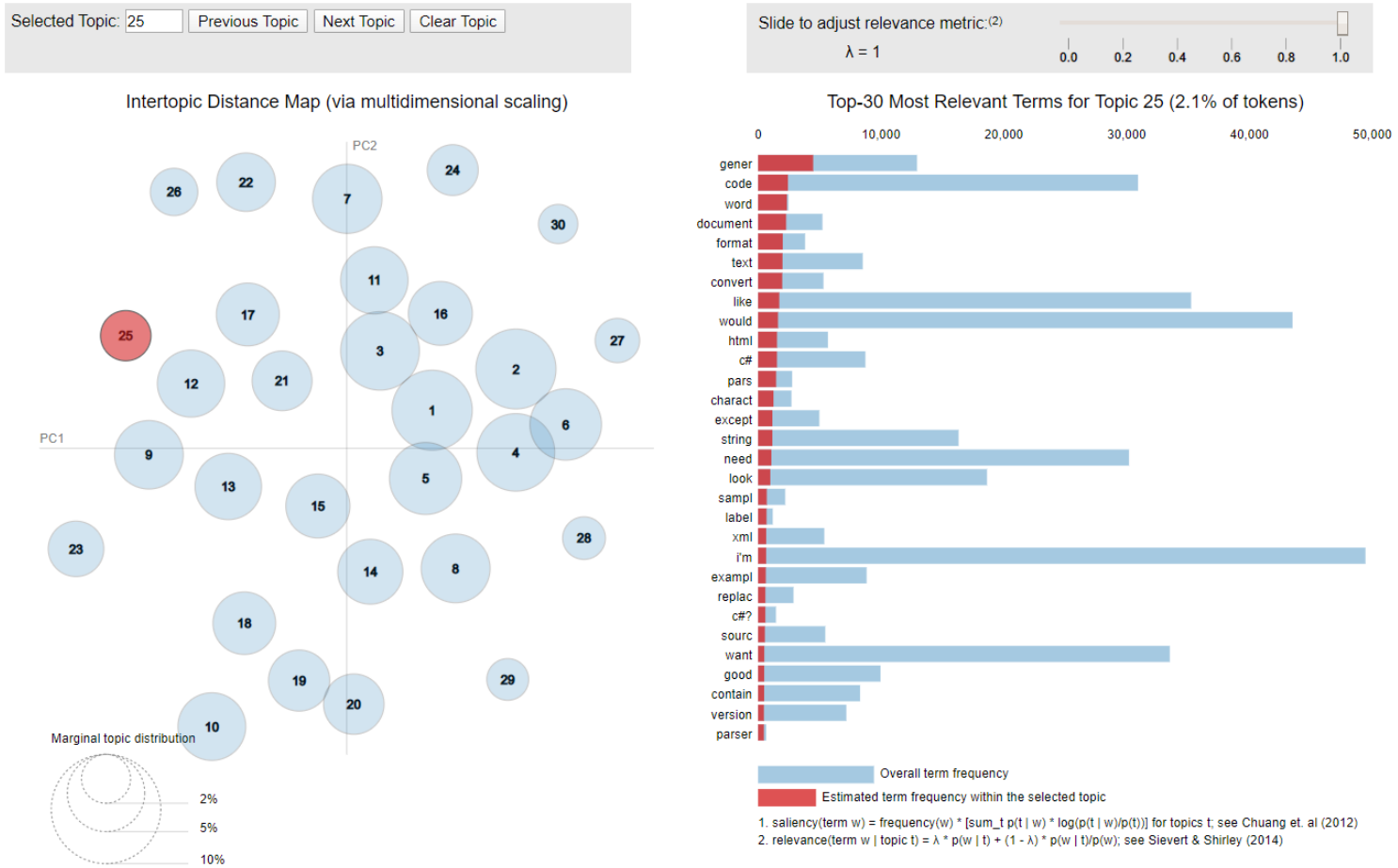
Données finales et prêtes à la modélisation :

data	signification	dimensions	format
df.csv	Tous les données brutes et transformées	(103 676, 12)	csv
X_C.npz	Matrices Bag of Words output **Count_vectorizer**	(103 676, 31541)	sparce
X_T.npz	Matrices output **TF-IDF**	(103 676, 31541)	sparce
Y.npz	Target output **MultiLabelBinarizer**	(103 676, 12403)	sparce

3. MODÉLISATION : APPROCHE NON SUPERVISÉE

La modélisation automatique de sujet :

- **LDA : Latent Dirichlet Allocation :** une méthode non-supervisée générative qui permet de détecter les sujets latents abordés dans un corpus de documents, trouver les mots les plus importants par sujet, et enfin assigner les sujets détectés aux différents documents.



Remarques :

- On choisi le nombre de sujets
- Evaluation : Cohérence
- Outil graphique interactif du package **pyLDAvis**

Conclusion :

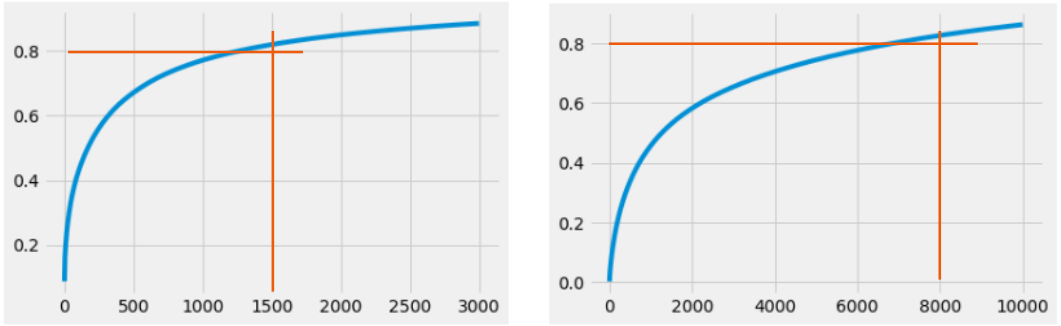
- Des Tokens « insignifiant »
- Des thèmes mélangés
- Temps de traitement considérable
- 12403 Tags : compliqué

➔ On passe à l'approche supervisée

4. MODÉLISATION : APPROCHE SUPERVISÉE

- **Réduction de dimensions : truncated SVD de sklearn :** une réduction de dimensionnalité linéaire au moyen d'une décomposition de valeur singulière tronquée

variance cumulée pour réduction de dimension BOW & TF_IDF



- 1500 variables de BOW qui représentent plus que 80% de la variance
- 8000 variables du TF_IDF qui représentent plus que 80% de la variance

Problème de classification multi labels :

- **Approche : combinaison des classifieurs :**
  - **One_versus_rest Classifier :** créer K classifieurs binaires qui séparent chaque classe k de l'union des autres classes
  - **Modèles utilisés pour classifieur interne :** Linear SVC, Random Forest, XGBoost, SGDClassifier

- **Evaluation :**
  - **Accuracy :** Le taux des prédictions correctes, « toujours faible »
  - **F1_score :** La moyenne harmonique de la précision et du rappel avec une extension f1_score(y_test, y_pred, average='micro')
  - **Hamming Loss :** la fraction des mauvaises étiquettes prédites par rapport au nombre total d'étiquettes
  - **Taux des predictions non nulles**

4. MODÉLISATION : APPROCHE SUPERVISÉE

- **Base Line Prediction** : Random Forest avec TF_IDF réduit à 100 variables

```
True tags:      ('date', 'datetime', 'javascript', 'time')
Predicted tags: ('javascript',)

True tags:      ('distro', 'java', 'linux')
Predicted tags: ('java',)

True tags:      ('database', 'php', 'security')
Predicted tags: ('php',)

True tags:      ('.htaccess', 'php', 'xml')
Predicted tags: ('php',)

True tags:      ('.net', 'web-services')
Predicted tags: ('java',)
```

```
hamming_loss(y_test, pred_val_tfidf_rf)
```

```
0.00023616887168662005
```

➔ **Erreur de jugement**

Modèle	Données à l'entrée	Temps d'entrainement	Temps de prédiction	Accuracy	Hamming Loss	% des prédictions nulles
Random Forest	BOW (103 676, 1500)	7h58min	46min	0.00054	0.00023	> 95%
Random Forest	TF_IDF (103 676, 8000)	18h37min	2h38min	0.00154	0.00023	> 95%

Tester les autres classifieurs :

- hyperparamètres de base
- Bag Of Words (1000 questions, 500 variables)
- Tuning manuel

➔ **Le SGDClassifier** donne des prédictions non nulles pour 70% et prend beaucoup moins de temps pour s'entrainer, c'est pour ça que c'est le modèle que j'ai choisi pour le tuning malgré ses performances un peu dépassées.

Modèle	Temps d'entrainement	Temps de prédiction	Accuracy	Hamming Loss	% des prédictions nulles
RF (10, 4)	1m18s	4m30s	0.00077	0.00096	99.3%
RF (10, 10)	1m20s	4m30s	0.00231	0.00097	93.7%
RF (50, 10)	4m47s	4m58s	0.00154	0.00096	97.1%
RF (50, 20)	4m57s	4m58s	0.00154	0.00096	97.2%
RF (50, 50)	4m57s	4m58s	0.00308	0.00096	96.6%
RF (100, 10)	9m29s	5m09s	0.00231	0.00096	96.9%
RF (100, 20)	9m25s	5m09s	0.00154	0.00096	96%
XGBoost ('gbtree')	4m02s	3s	0.00462	0.00095	87.5%
XGBoost ('dart')	4m21s	3s	0.00462	0.00095	87.5%
XGBoost (' gblinear')	3m17s	3s	0	0.00096	100%
LinearSVC	1m04s	0.4s	0.00925	0.00098	78.1%
SGDClassifier	15s	0.4s	0.00694	0.00148	30%

4. MODÉLISATION : APPROCHE SUPERVISÉE – TUNING ET MODÈLE FINAL

- **Le GridSearch** : TF_IDF(1000, 8000), SGDClassifier :

```
svdt = TruncatedSVD(n_components=8000)
XT_T = svdt.fit_transform(X_T)
```

```
grid = {
    'estimator__alpha': [0.000001, 0.00001, 0.0001, 0.001],
    'estimator__n_iter': [1, 10],
    'estimator__penalty': ['l1', 'l2']
}
```

- **Résultat :**
  - **Penalty, terme de régularisation :** l2
  - **Alpha, cste :** 0.00001
  - **N_iter :** 1

```
model_tunning.best_params_
```

```
{'estimator__alpha': 1e-05, 'estimator__n_iter': 1, 'estimator__penalty': 'l2'}
```

- **Modèle final :**
  - **Données** : TF_IDF de dimensions (103 676, 8000) représentant plus que 80% de la variance
  - **Modèle :** SGDClassifier (alpha = 0.00000, penalty = « l2 »)
  - **Temps d'entrainement :** 28h52min
  - **Temps de prédictions de toutes les questions de test (25000):** 1h34min
  - **Accuracy :** 0.07307
  - **Hamming Loss :** 0.000201
  - **F1_Score :** 0.46
  - **% des prédictions nulles :** 12.6%

	Tags prédites	*Vraies Tags*
0	(crystal-reports,)	(asp-classic, sql-server, crystal-reports)
1	(comments, emacs)	(emacs, elisp)
2	(c++, file)	(bulkinsert, iostream, file-io, c++)
3	(visual-studio, visual-studio-2008)	(visual-studio,)
4	(asp.net, email, php)	(php, email-spam, email)
5	(forms, html, xhtml)	(html, standards-compliance)
6	(app-store, iphone)	(iphone,)
7	(c++, winapi, windows)	(winapi, sdk, c++)
8	(caching,)	(cpu-cache, performance, cpu-architecture, intel)
9	()	(sql-server, string, sql)
10	(php, regex)	(php, regex)
11	(visual-studio, visual-studio-2008)	(visual-studio,)
12	(ruby,)	(ruby, file-io)

5. DÉPLOIEMENT DE L'API

Flask :

- Flask est un framework open-source de développement web en Python.

Serverless :

- Le Serverless Framework est un framework web gratuit et open-source écrit en utilisant Node.js. Serverless est le premier framework développé pour créer des applications sur AWS Lambda et gérer la notion Infrastructure As A Code.

Git :

- Le code est mis à disposition sur un repository github :

  https://github.com/amcharek/OCP6.git

5. CONCLUSION ET PERSPECTIVES

- On note qu'il y'a une marge d'amélioration :
  - La partie nettoyage et traitements des tokens
  - On peut se limiter aux tags qui reviennent plus que 10 fois
  - On peut creuser la partie non supervisée en cherchant un nombre optimal de sujet

- En traitant ce sujet j'ai dû acquérir des nouvelles compétences en :
  - Traitement de langage naturel : **NLP**
  - Prétraitement des données non structurées pour obtenir un jeu de données exploitable
  - Méthode et algorithme de classification multi labels
  - Méthode non supervisée pour le traitement de texte
  - Utiliser un logiciel de gestion de versions de code, Git
  - Infrastructure As A Code, serverless, AWS : Lambda, S3 & API Gateway

Merci de votre attention

Ayoub MCHAREK

ayoub.mcharek@outlook.com