

# CATÉGORISATION DE PRODUITS

## ATELIERS-BIG-DATA

---

Brendan Guillouet

11 Décembre 2017

Institut National des Sciences Appliquées

Introduction

Nettoyage des données

Vectorisation

Apprentissage

TP

# INTRODUCTION

---

Multiple **DOMAINE DE RECHERCHE** et une très grande variétés d'**APPLICATIONS** :

- Recherche d'information
  - *Moteur de recherche (Google, Yahoo)*
- Traitement Automatique du langage
  - *ChatBot*
- Reconnaissance de pattern
  - *Scrawling de page*
- Analyse de sentiment
  - *Marketing*
- Désambiguïsation
  - *Sécurité*

## OBJECTIF :

Automatiser la catégorisation des produits dans l'arborescence du site. (*Concours Datascience.net*)

<https://www.datascience.net/fr/challenge/20/details>

- 47 Catégories de niveau 1.
- 536 Catégories de niveau 2.
- 5789 Catégories de niveau 3.

## DIFFICULTÉS :

- Gérer des gros volumes de données (**Big Data**) :
  - *dizaines de millions de produits.*
- Données **réelles** :
  - *étape de nettoyage importante.*
- Appréhender un nouveau type de données : **textuelles**.

Données issues du concours proposé par **Cdiscount** et disponible sur **Datascience**.

Fichier d'apprentissage de 15.786.885 produits.

Champ	Type de données	Description
Identifiant produit	String	Identifiant unique du produit
Catégorie 1	Int	Catégorie de niveau 1
Catégorie 2	Int	Catégorie de niveau 2
Catégorie 3	Int	Catégorie de niveau 3
Description	String	Description produit
Libelle	String	Description courte
Marque	String	Marque du produit

TABLE 1 – Données Cdiscount

Catégorie1	Description
CULTURE - JEUX	playstation 4 et jeu pes 4
CULTURE - JEUX	xbox 360 et call of duty
CULTURE - JEUX	manette pour xbox et playstation 4
JARDIN - PISCINE	grande pelle verte
JARDIN - PISCINE	pompe a eau pour jardin
AUTO - MOTO (NEW)	360 casque avec visière
AUTO - MOTO (NEW)	pompe a eau et gaz

**TABLE 2** – Exemple de catalogue d'apprentissage

## DIFFICULTÉS :

- **Bruits** liés aux fautes d'orthographe, aux accords, à la conjugaison.
- Nombreux termes **non significatifs**.
- Termes significatifs liés **au contexte**.
- **Transcription** aux outils machine learning.
- Décision liée au **contexte**.
- Traitement **différent** d'une langue à l'autre.

⇒ Étape de pré-traitement des données très importante.



# NETTOYAGE DES DONNÉES

---

**PROBLÈME :** Des termes peuvent-être écrits de **différentes façons** (accents, conjugaison, genre, pluriels..) et néanmoins avoir le **même sens**.

**SOLUTION :** Remplacer les mot par leur **racine**.

**EXEMPLE :**

- épée, épee, epée = epe
- vert, verts, verte, vertes = vert
- mange, manger, mangez, mangent = mang

De nombreux algorithmes différents propre aux langues étudiés  
Algorithme utilisé *Snowball (Porter)*

## PROBLÈME :

Des termes très usuels, donc non différenciateurs, peuvent perturber l'apprentissage.

## SOLUTION :

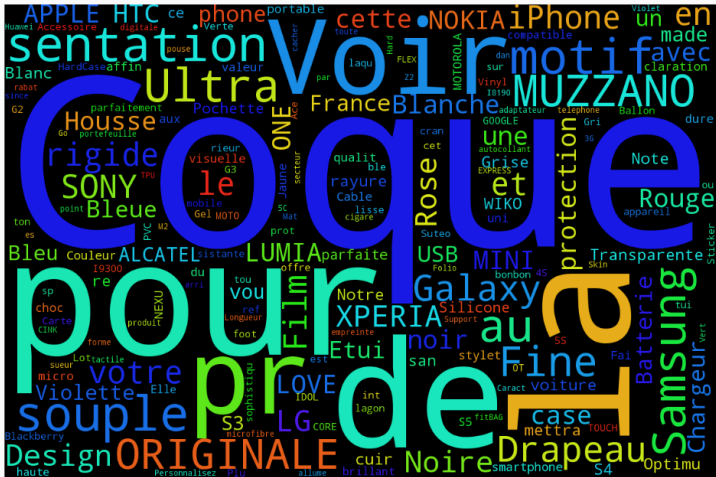
Lors de l'étape de preprocessing, les mots les plus communs sont supprimés à partir d'une liste de **Stopword** :

(["a", "afin", "ai", "ainsi", "après", "attendu", "au", "aujourd", "auquel",  
"aussi", "autre", "autres", "aux", "auxquelles", "auxquels", "avait", "avant",  
"avec", "avoir", "c", "car", "ce", "ceci", "cela", "celle", "celles", "celui",  
"cependant", "certain", "certaine", "certaines", "certains", "ces", "cet", "cette",  
"ceux", "chez", "ci",])

⇒ Également propre à la langue !

- Suppression de code HTML (*BeautifulSoup*).
- Suppression de la ponctuation.
- Incrémenter les stopwords avec de la connaissance métier.
- Suppression des caractéristiques numériques (Sauf pour les marques)

## ILLUSTRATION - CATÉGORIE "TÉLÉPHONIE - GPS"



Mots les plus fréquents avant nettoyage.





- **NLTK** (*Python, Spark*) : Traitement du langage (racinisation, stopwords, ...).
- **Lucène** (*Java*) : Librairie d'indexation et de recherche de texte.
- **BeautifulSoup** : Suppression des balises HTML.
- **Regex** : Langage de recherche de texte.



# VECTORISATION

---

- Transformer la liste de mots sous un format **interprétable** par les différents algorithmes d'apprentissage.
- Gérer le très **grand nombre** de features.
  - Exemple sur 21.543 lignes de la catégorie "TÉLÉPHONIE-GPS"
  - 24.486 mots uniques -> 8.384 après nettoyage.
- Choisir des poids **significatifs**.

**Vectoriser** les descriptions tout en **réduisant** l'espace de stockage.

$X$

$\Rightarrow$

$\phi$

Vecteurs de tailles aléatoires.

Très grand nombre de features.

Taille inconnue

Vecteurs de tailles fixes.

Taille fixé à l'avance : **n\_hash**.

- Éviter les collisions.
- Accélérer la vitesse de calcul.
- Non aléatoire.

## Definition (Fonction de hashage - 1)

$$h: \mathbb{N} \rightarrow \{1, \dots, n\_hash\}$$
$$i \mapsto j = h(i).$$

## Definition (Fonction de hashage - 2)

$$\xi: \mathbb{N} \rightarrow \{1, -1\}$$
$$i \mapsto j = \xi(i).$$

## Definition (Hashed Feature Map)

$$\phi_j^{\xi, h}(x) = \sum_{i \text{ s.t. } h(i)=j} \xi(i)x_i$$

## PROBLÉMATIQUE :

Certains mots n'ont pas le même sens en fonction du contexte.

- *Short de bain*  $\neq$  *Short*  $\neq$  *bain*

## SOLUTION :

On ne considère pas que les mots uniques (*unigram*) mais les couples de mots (*bigram*), ou toutes associations de  $n$  mots ( $n$ -gram).

- Résout les ambiguïté du langage.
- Explosion de la taille des vecteurs
  - Exemple sur 21.543 lignes de la catégorie "TELEPHONIE-GPS"
  - 8.384 *unigram*, 50.012 *bigram*, 90.854 *trigram*...

Assigner un **score d'importance** d'un mot, ou d'une association de mot, dans un document **relativement à un ensemble de document**.

- $t$  : mot ou association de mot.
- $d$  : un document.
- $D$  : un ensemble de document.

## Definition (Formule Générale TF-IDF)

$$tfidf(t, d) = tf(t, d) \times idf(t, D)$$

- $tf(t, d)$  : *Term-Frequency* Nombre d'occurrence du terme  $t$  dans le document  $d$ .
- $idf(t, D)$  : *Inverse-Document-Frequency* Mesure l'importance du terme  $t$  dans l'ensemble des documents  $D$ .

## FORMULE DU TF

Le terme  $tf(t, d)$  est généralement défini comme le nombre d'occurrences du terme  $t$  dans le document  $d$  :

$$tf(t, d) = f_{t,d}$$

Cette définition est celle utilisée dans les librairies *scikit-learn* de *Python* et *Mllib* de *Spark*.

Cependant des variations peuvent exister :

binaire	0, 1
normalisation logarithmique	$1 + \log(f_{t,d})$
normalisation « 0.5 » par le max	$0.5 + 0.5 \times \frac{f_{t,d}}{\max_{t' \in d} f_{t',d}}$
normalisation par le max	$K + (1 - K) \times \frac{f_{t,d}}{\max_{t' \in d} f_{t',d}}$

**TABLE 3** – Variante du TF (source wikipedia)

La formule du terme *IDF* varie également d'une implémentation à l'autre.

$\log(\frac{N_D}{DF(t,D)})$	
$\log(\frac{N_D+1}{DF(t,D)+1})$	<i>MLib (Spark)</i>
$\log(\frac{N_D+1}{DF(t,D)+1}) + 1$	<i>scikit-learn (Python)</i>

**TABLE 4** – Variante de l'IDF

- $N_D$  : Nombre de documents.
- $DF(t, D)$  : Nombre de documents dans lequel le terme  $t$  apparaît.



- Les fonctions de *Hashage* et de *TF-IDF* sont appliquées sur le jeu de données d'apprentissage.
- La même fonction de *Hashage* est utilisée sur le jeu de données test.
- Les termes de *TF* entre un mot  $t$  et un document  $d$  sont recalculés pour le jeu de données test
- Les termes d'*IDF* calculés lors de l'apprentissage sont réutilisés.

# APPRENTISSAGE

---

- Par défaut, *scikit-learn*, *MMLib* utilisent des algorithmes "one-vs-all"
- Dans *scikit-learn* différents solveurs sont disponibles (*liblinear*, *LBFGS*, *sag*...).

- Les 4 meilleures solutions ont utilisées des méthodes linéaires Goutorbe et al. [2016]
- Solutions à plusieurs niveaux.
- Implémentation de la solution N°2 disponible : <https://github.com/ngaude/cdiscount>

TP





### RÉFÉRENCES

---

Bruno Goutorbe, Yang Jiao, Matthieu Cornec, Christelle Grauer, and Jérémie Jakubowicz. A large e-commerce data set released to benchmark categorization methods. 2016.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1113–1120. ACM, 2009.