

Text Mining

Présenté par : Pr. Ikram EL Asri
Contact : elasri@inpt.ac.ma

Objectifs pédagogiques

- Comprendre les principes de fonctionnement du text mining
- Mettre en œuvre le traitement d'un corpus par les méthodes de statistique textuelle
- Comprendre les principes de fonctionnement des principaux algorithmes de classification
- Interpréter les résultats
- Comprendre les compétences nécessaires pour ce type de projet et la nécessité d'adopter des méthodes de déploiement spécifiques à ces types de projet.

Structure du cours

- **Cours :**
- 7 séances de cours
- Inclus des exemples pratiques et des tps qui facilitent la compréhension des cours magistraux et ajoutent du matériel pratique
 - Le langage de programmation principal est Python
- Supports de cours sont disponible sur la plateforme moodle

Evaluations :

- Remise des exercices
- Réalisation Projet

Data Mining

Plan du cours :

Introduction au Text Mining

- Du data mining au text mining
- Applications du Text Mining

La matrice documents termes

- Représentation Bag of Words
- Réduction de la dimensionnalité
- Pondération
- Mesurer les similarités

La catégorisation de textes

Topics model

Fouille d'opinions et analyse des sentiments

Détection des communautés

Filtrage collaboratif et recommandations

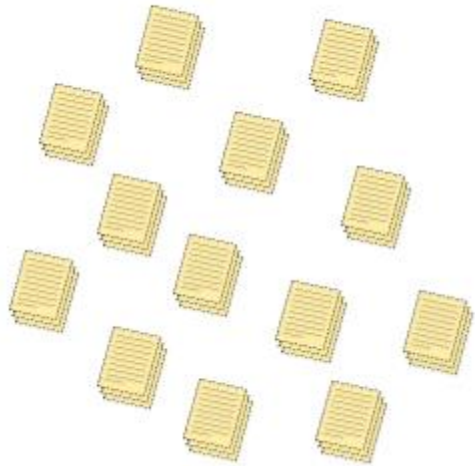
Introduction au Text Mining

DU DATA MINING AU TEXT MINING

APPLICATIONS DU TEXT MINING

Du data mining au text mining

Le data mining est un processus d'extraction de structures (connaissances) inconnues, valides et potentiellement exploitables dans les bases (entrepôts) de données (Fayyad, 1996), à travers la mise en œuvre des techniques statistiques et de machine learning.



Les données textuelles constituent également une source d'information qui permettrait **d'extraire de la connaissance** (détecter des régularités [patterns], **recherche des similarités**, **identifier les relations de causalité**, etc.).

Du data mining au text mining

Text mining – Spécification

- Text mining cherche à extraire des informations utiles d'une collection de documents.
- C'est similaire Data Mining , mais les sources de données sont des documents non structurés ou semi-structurés.
- Text mining exploite des techniques / méthodologies issues de **l'exploration de données**, de **l'apprentissage automatique**, de **la recherche d'informations**, de **la linguistique computationnelle basée sur un corpus**

Texte = document non structuré

- **Document structuré**

- Les données usuellement exploités en data mining se présentent sous forme de tableaux attributs-valeurs. Ligne = individu statistique, colonne = attribut (descripteur). Les algorithmes sont taillées pour les traiter.

sep_length	sep_width	pet_length	pet_width	type
5.1	3.8	1.6	0.2	setosa
5.8	2.7	5.1	1.9	virginica
4.9	2.4	3.3	1.0	versicolor
7.7	2.6	6.9	2.3	virginica
6.6	3.0	4.4	1.4	versicolor
6.2	2.8	4.8	1.8	virginica
5.0	2.3	3.3	1.0	versicolor
4.4	3.0	1.3	0.2	setosa
5.7	2.8	4.5	1.3	versicolor

- **Document non structuré**

- Les données en text mining se présentent sous forme de textes bruts. Les algorithmes de data mining ne savent pas les appréhender nativement.

<texte>

Resdel Industries Inc said it has agreed to acquire San/Bar Corp in a share-for-share exchange, after San/Bar distributes all shgares of its Break-Free Corp subsidiary to San/Bar shareholders on a share-for-share basis.

The company said also before the merger, San/Bar would Barry K. Hallamore and Lloyd G. Hallamore, San/Bar's director of corporate development, 1,312,500 dlrs and 1,087,500 dlrs respectviely under agreements entered into in October 1983.

</texte>

Du data mining au text mining

Texte = document pas si non-structuré que cela

Cadre pour la catégorisation de textes multilingues

Radwan JALAM, Jérémy CLECH, Ricco RAKOTOMALALA
{jalam, jclech, rakotoma}@eric.univ-lyon2.fr

Laboratoire ERIC
Université Lumière Lyon 2
5, av. Pierre Mendès-France
69676 Bron, France
fax: (33) 4 78 77 23 75

Abstract

In this paper, we propose an original framework for multilingual text categorization. The objective is to classify a set of texts, written in some language, using a predictive model learned from a set of texts written in a given language, called learning language. Contrary to the unilingual classical phase of text categorization, the classification phase contains two new steps : firstly identify the language of the text, and then automatically translate it into the learning language. As shown in this paper, first applications of multilingual text categorization on real data, that is over English, French and German newspapers, indicate that the approach is viable.

Résumé

Dans cet article, nous proposons un cadre pour la catégorisation de textes multilingues. L'objectif est de pouvoir, à partir d'un modèle de prédiction construit par apprentissage sur un corpus de textes rédigés dans une langue donnée, inférer sur une série de textes qui sont rédigés dans une langue quelconque. Cette phase d'inférence, par rapport à la généralisation classique, comprend deux étapes supplémentaires : la détection de la langue du texte, puis sa traduction automatique vers la langue de référence. Nos premiers résultats sur une application réelle : la catégorisation d'articles de journaux allemands, anglais et français, montrent la viabilité de l'approche.

Keywords: Multilingual text categorization, Language identification, Machine learning, n-grams representation, CLEF collection, Translation effects on text categorization.

1 Introduction

La catégorisation de texte consiste à chercher une liaison fonctionnelle entre *un ensemble de textes* et *un ensemble de catégories* (étiquettes, classes). Cette liaison fonctionnelle, que l'on appelle également *modèle de prédiction*, est estimée par un apprentissage automatique (traduction de *machine learning method*). Pour ce faire, il est nécessaire de disposer d'un ensemble de textes préalablement étiquetés, dit *ensemble d'apprentissage*, à partir duquel nous estimons les paramètres du modèle de prédiction le plus performant possible, c'est-à-dire le modèle qui produit le moins d'*erreurs* en prédiction (voir la figure 1).

Formellement, la catégorisation de texte consiste à associer une valeur booléenne à chaque paire $(d_j, c_i) \in \mathcal{D} \times \mathcal{C}$, où \mathcal{D} est l'ensemble des textes et \mathcal{C} est l'ensemble des catégories. La valeur V (Vrai) est alors associée au couple (d_j, c_i) si le texte d_j appartient à la classe c_i tandis que la valeur F (Faux) lui sera associée dans le cas contraire. Le but de la catégorisation de texte est de

Titre

On parle de document non structuré au sens du data mining, cela ne veut pas dire qu'un document n'obéit pas à une certaine organisation. *Ex. article scientifique.*

Résumés

Mots-clés

Organisation
en sections,
paragraphe.

Du data mining au text mining

- Nécessité du text mining

- **L'analyse textuelle est un problème ancien** → L'exemple le plus simple (et ancien) est le traitement des questions ouvertes dans les enquêtes.
- **Text mining une nécessité de plus en plus patente avec la profusion des documents au format numérique** → Profusion des documents, augmentation des capacités de stockage, apparition de nombreux corpus spécialisés.
- **Text mining incontournable avec l'explosion du web** → Le web 2.0 a démultiplié dans des proportions gigantesques la disponibilité de documents rédigés, stockés numériquement (ex. newsgroups, réseaux sociaux [twitter], forums, etc.). Les opportunités d'extraction de connaissances utiles sont infinies.

Processus Text Mining

La préparation des données va jouer un rôle fondamental parce que les techniques statistiques usuelles ne sont pas armées pour traiter de la donnée non structurée.

Collection de documents
= **Corpus** = Base
d'apprentissage

Bag Of Words

Simple counting
Statistics

Mapping/Visualization
Result interpretation



Text

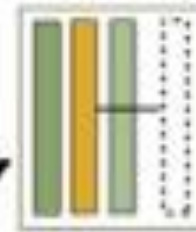
Text Preprocessing



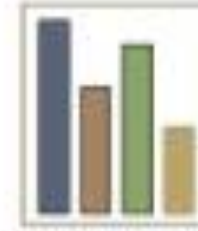
Text Transformation
(Feature Generation)



Feature Selection



Data Mining /
Pattern Discovery



Interpretation /
Evaluation



APPLICATIONS DU TEXT MINING

Text mining est utilisé dans plusieurs applications :

1. Catégorisation de textes
2. Clustering de textes
3. Recherche d'information
4. Extraction d'information
5. D'autres applications populaires

APPLICATIONS DU TEXT MINING

- *Catégorisation de textes*

- Cadre de l'apprentissage supervisé : exploiter une collection de documents préalablement étiquetés

Document Classe



+
+
-
-
+



Construction d'un modèle
prédictif



$\Phi(\text{document})$

Qui peut être utilisé pour l'étiquetage d'un
nouveau document

APPLICATIONS DU TEXT MINING

- *Catégorisation de textes : Remarques*

- On parle de « document classification » en anglais
- Le processus permet également d'associer les documents à des termes prédéfinis, on parle alors d'indexation automatique
- Le cadre le plus fréquent est binaire avec une classe d'intérêt (« + ») contre les autres (« - ») qui peuvent être constitués de documents très hétérogènes)
- Une application phare est le filtrage automatique de documents (ex. e-mail, pages web, etc.) ; mais il y en a d'autres (analyse des sentiments, opinion mining, etc.)

APPLICATIONS DU TEXT MINING

Catégorisation de textes : Exemple : la détection de SPAM

Une base d'apprentissage constituée de documents SPAM et NON-SPAM permet de construire un modèle prédictif.

Email



Machine Learning Model

Ce modèle issu du processus data mining est déployé sur le serveur. Il attribue un score aux e-mails entrants. Ceux qui présentent un score trop élevés sont bloqués.

Spam



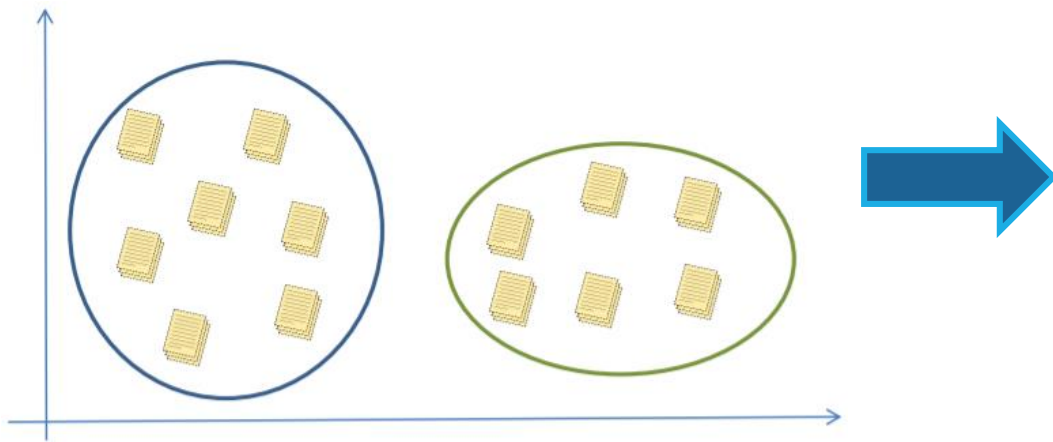
Not Spam



APPLICATIONS DU TEXT MINING

- *Clustering de textes*

- Cadre de l'apprentissage non-supervisé : partitionner en groupes homogènes les documents



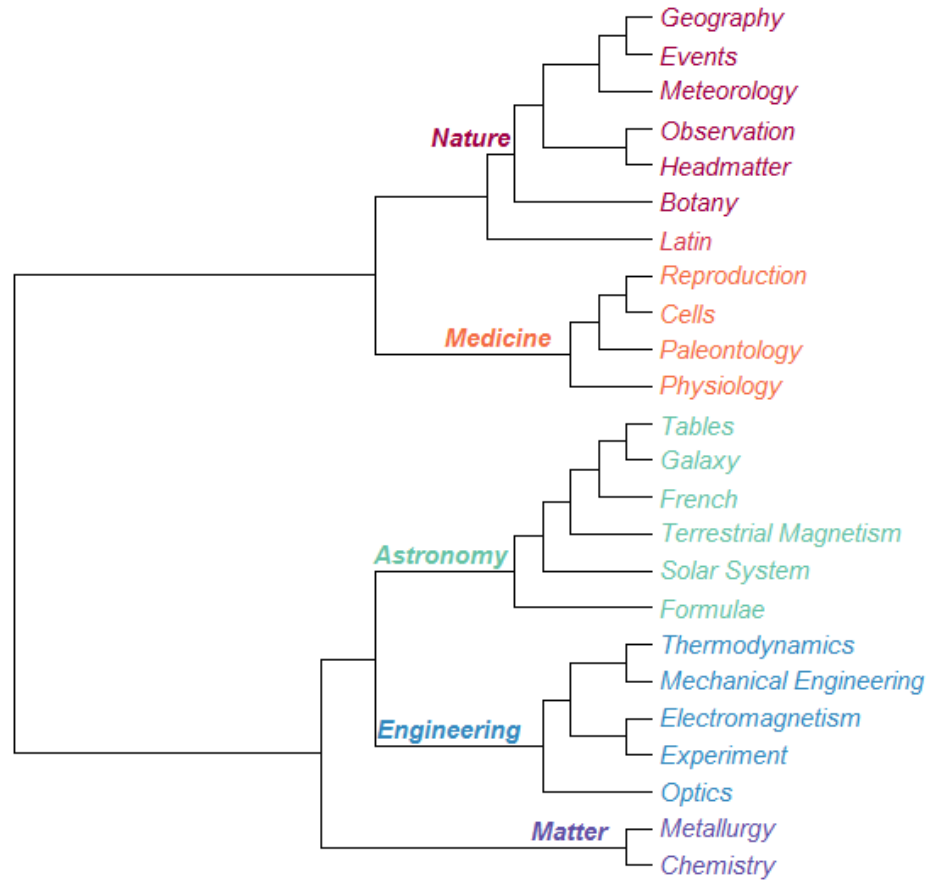
Les documents d'un même groupe sont le plus similaires possible. Les documents de groupes différents sont le plus dissimilaires possible.

On bénéficie des avantages liés à la structuration : une organisation des documents pour faciliter la consultation et la recherche ; disposer d'un résumé du corpus (une forme de réduction de la dimensionnalité) ; etc.

L'enjeu, toujours en clustering, est de comprendre la nature des groupes, en les associant à des thèmes par exemple.

APPLICATIONS DU TEXT MINING

- *Clustering de textes - Exemple*



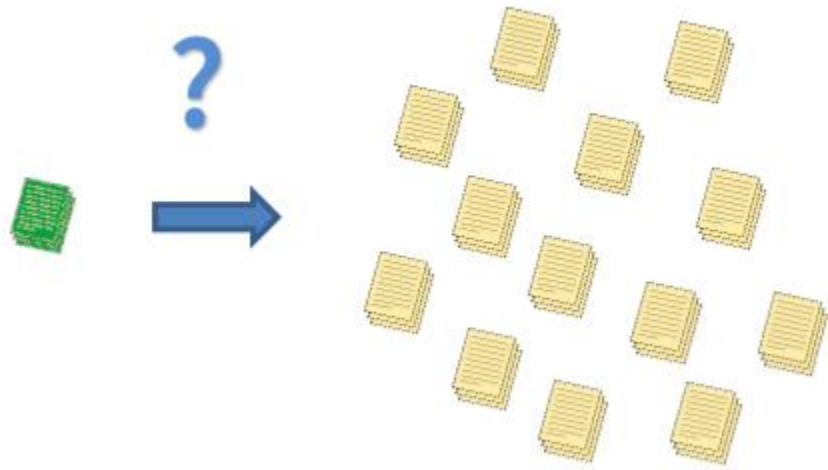
Topics that typically co-occur in documents have similar topic-document distributions

a selection of five topics with the most pronounced
: *Nature, Latin, Medicine, Astronomy, Engineering, Matter*

Hierarchical clustering of topics by their topic-document distribution

APPLICATIONS DU TEXT MINING

- *Recherche d'information*



L'objectif de la recherche d'information (*information retrieval*) est de mettre en place les stratégies permettant d'identifier, dans un corpus, les documents pertinents relatifs à un document requête.

Il s'agit d'une recherche par le contenu, le texte est concerné, mais elle peut s'étendre à l'image, la vidéo, le son...

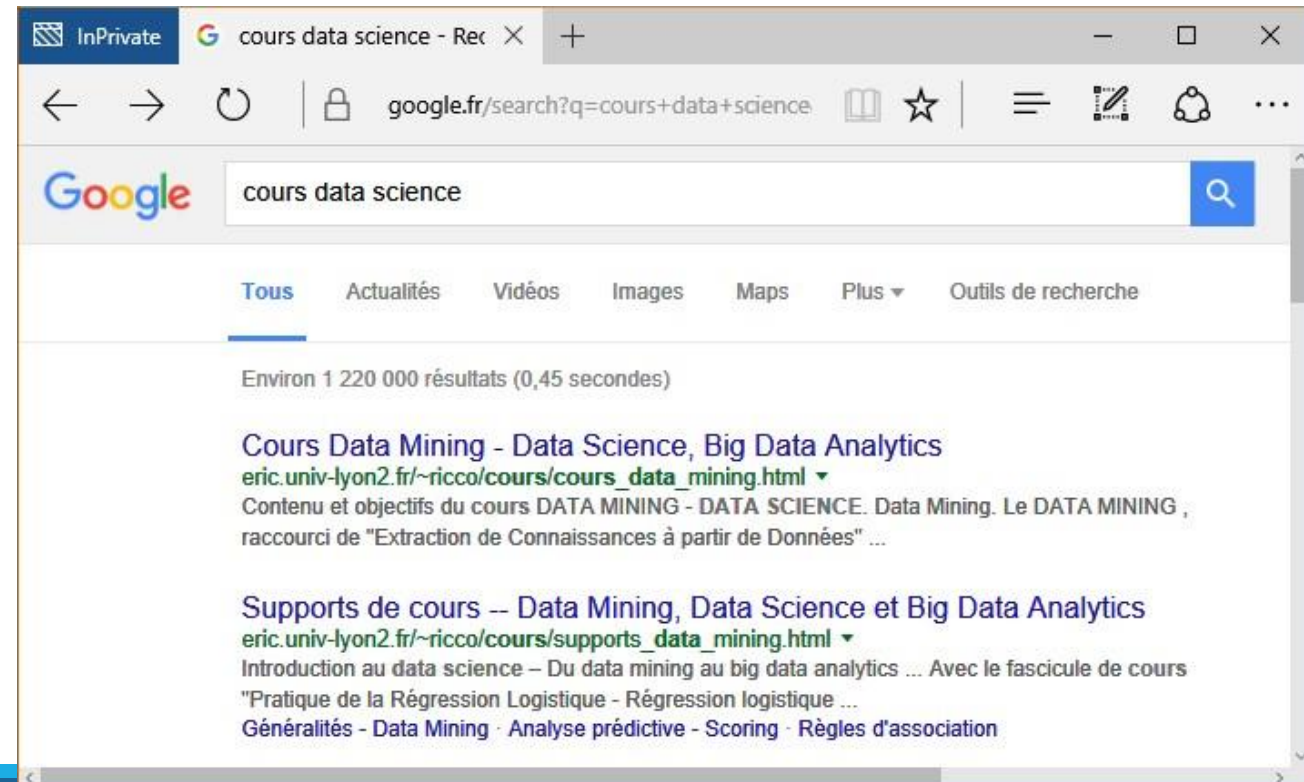
APPLICATIONS DU TEXT MINING

- *Recherche d'information*

- L'indexation joue un rôle fondamental dans la recherche, la mesure de similarité entre documents...
- Les applications sont très nombreuses. Ex. détection de plagiat, recherche de dossiers médicaux, etc.
- Il existe des bibliothèques logicielles performantes (Ex. Lucene, Elasticsearch).

APPLICATIONS DU TEXT MINING

- *Recherche d'information : exemple particulier des moteurs de recherche web*
- Les moteurs de recherche web constituent l'exemple le plus emblématique de la **recherche d'information**:
 - le document requête est très court,
 - constitué de quelques mots clés ;
 - des stratégies de spamming viennent polluer la recherche



APPLICATIONS DU TEXT MINING

- *Extraction d'information*

- L'extraction d'information consiste à rechercher **des champs prédéfinis** dans un texte plus ou moins rédigé en langage naturel. On s'appuie plus sur l'analyse lexicale et morphosyntaxique pour identifier les zones d'intérêts.

Annonce 1

Description :
FIAT COUPE 2L 20V 155 CV
1999
136000 km
Contrôle technique ok
Prix: 2990 €
Equipement :
Cuir. Climatisation. Vitres électriques. Rétroviseurs électriques. Direction assistée. Fermeture centralisée. ABS. Airbag. Jantes aluminium

Annonce 2

Fiat coupe T 16 (moteur lancia delta dorigine) an 95
tuning leger rabaisse av et ar blistein b8 ressort court ct
ok en cours le vehicule roule tres peu (suite perte emploi
) il es entretenu marche tres fort environ 200 ch il peut
parcourir tous les trajet. je ne suis pas presse alors les
marchand de tapis pas la peine de venir la peinture a ete
refait par lancien propriétaire ainsi que lembayage mais
elle as des imperfections demande de renseignement par
tel uniquement

Ex. extraire automatiquement les informations additionnelles sur les annonces automobiles.

Annonce	2L	5 cylindres	Turbo	Bolidée	1ère main	Réparations à prévoir
1	oui	oui	non	non	?	non
2	oui	non	oui	oui	?	oui
3

Objectif : Récupérer automatiquement les informations qui ne sont pas standardisées sur leboncoin.fr (ex. pour les voitures : année modèle, kilométrage, carburant, boîte)

APPLICATIONS DU TEXT MINING

- *D'autres applications populaires*

Résumé automatique
(Automatic
summarization)



Résumé d'un document. Recherche des phrases les plus représentatives dans un document.
Résumé d'un corpus. Recherche du document le plus représentatif dans un corpus ou Recherche des phrases représentatives à partir de plusieurs documents.
La notion de distance entre phrases, entre documents, joue un rôle important.

Identification des
tendances

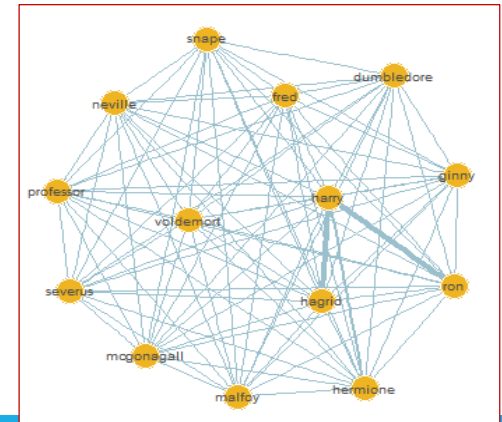


Les données arrivent en flux. S'appuyer sur la temporalité (ex. textes sur les réseaux sociaux). Analyser l'évolution de la popularité des thèmes, identifier les sujets émergents (topic detection).

Analyse des liens



Analyser les relations (cooccurrences) entre les termes, qui peuvent être des personnages (nous approfondirons cette idée dans l'analyse des réseaux sociaux)



La matrice documents termes

REPRÉSENTATION BAG OF WORDS

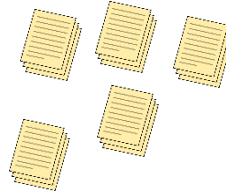
RÉDUCTION DE LA DIMENSIONNALITÉ

PONDÉRATION

Notions Principales



Document = individu
statistique



« Base » d'apprentissage = collection de
documents = **Corpus**

Enjeu : traduire la collection de documents en un tableau de données « attributs-valeurs » propice au traitements à l'aide des algorithmes de data mining, **en minimisant au possible la perte d'information.**



N°	V1	V2	V3	...
1				
2		?		
3				
4				
5				

1. Que mettre en colonnes (V1, V2,...) ?
2. Quelles valeurs mettre dans le tableau ?

Représentation BAG OF WORDS

Représentation BAG OF WORDS

- BAG-OF-WORDS (Sac de mots) : La transformation d'une collection de textes en un tableau de données sans nécessiter de connaissances particulières sur le domaine étudié

The Bag of Words Representation

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



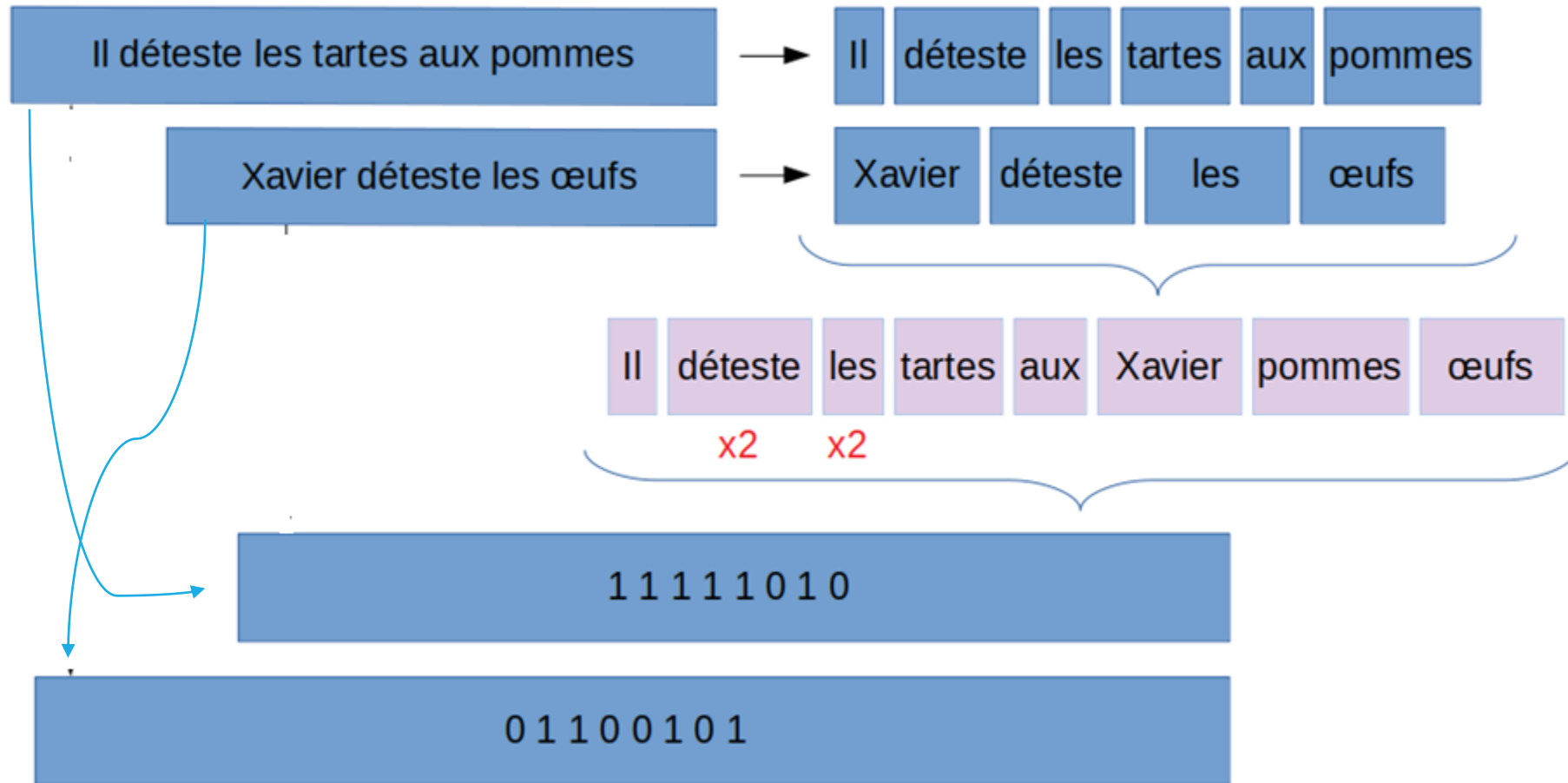
it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1

Représentation BAG OF WORDS

- **Etales :**
- 1. Repérer les mots (tokens) présents dans les documents
 - L'identification du délimiteur de mots est important. Ce sera souvent l'espace, les ponctuations, ... Certains caractères sont moins évidents (ex. « - » dans « écart-type »)
- 2. Qui vont constituer le dictionnaire
 - Potentiellement, le nombre de mots est très important. Il peut y avoir des redondances dans le dictionnaire, il faudra pouvoir les traiter (parfois évidentes : voiture vs. voitures ; parfois moins : voiture vs. automobile...).
- 3. Les mots deviennent des descripteurs (features, termes)
- 4. On associe alors l'absence ou la présence des mots à chaque document
 - Ca peut être aussi le nombre d'apparition des mots, ou autre type de valeurs. On parle de pondération.

Représentation BAG OF WORDS

- **Exemple 1 :**



Représentation BAG OF WORDS

- **Exemple 2 :**

1. Imaging databases can get huge
2. Most imaging databases save images permanently
3. Imaging databases store images.
4. Imaging databases store images. Imaging databases store images. Imaging databases store images



Document	imaging	databases	can	get	huge	most	save	images	permanently	store
1	1	1	1	1	1	0	0	0	0	0
2	1	1	0	0	0	1	1	1	1	0
3	1	1	0	0	0	0	0	1	0	1
4	1	1	0	0	0	0	0	1	0	1

BAG-OF-WORDS

- Fréquence des mots – **Word cloud**

Word cloud. Les WordClouds (nuages de mots-clés en français) sont des outils utiles pour synthétiser les notions les plus importantes d'un texte, d'une page web ou encore d'un livre. Plus un mot est présent dans le texte pris en considération, plus il apparaît en gros dans le wordcloud.

Il existe de nombreux sites et applications pour générer des wordclouds facilement et gratuitement.



Remarque : ‘de’, ‘la’, ‘le’... ont une importance démesurée dans notre wordcloud, on se rend compte que le nettoyage du texte est primordial.

Représentation BAG OF WORDS



Document	imaging	databases	can	get	huge	most	save	images	permanently	store
1	1	1	1	1	1	0	0	0	0	0
2	1	1	0	0	0	1	1	1	1	0
3	1	1	0	0	0	0	0	1	0	1
4	1	1	0	0	0	0	0	1	0	1

Remarques :

- Certains mots ne sont pas intrinsèquement porteurs de sens (ex. most) ;
- Certains mots véhiculent la même idée (ex. imaging vs. images) ou sont synonymes (ex. store vs. save)
- La pondération joue un rôle important (ex. n°3 et n°4 sont strictement identiques au sens de la présence/absence des mots, indûment ?)
- La question de la mesure de similarité/dissimilarité entre les documents sera centrale

RÉDUCTION DE LA DIMENSIONNALITÉ

La matrice documents termes

Représentation en sac de mots : la Tokenisation

Processus de « tokenization » : découpage d'un flux de caractères en phrases, symboles, mots. Nous nous intéressons plus particulièrement aux mots.

1. Imaging databases can get huge
2. Most imaging databases save images permanently
3. Imaging databases store images.
4. Imaging databases store images. Imaging databases store images. Imaging databases store images

Document	imaging	databases	can	get	huge	most	save	images	permanently	store
1	1	1	1	1	1	0	0	0	0	0
2	1	1	0	0	0	1	1	1	1	0
3	1	1	0	0	0	0	0	1	0	1
4	1	1	0	0	0	0	0	1	0	1

- La ponctuation « . » a été retirée.
- « » a joué le rôle de délimiteur.
- 4 documents → 4 lignes dans le tableau
- Le dictionnaire est composé des termes présents dans le corpus : {*imaging, databases, can, get, huge, most, save, images, permanently, store*}.
- La taille du dictionnaire n'est pas connue à l'avance.
- Le nombre de valeurs non nulles dans la ligne dépend de la longueur du document.

La matrice documents termes

Documents

Termes

Document	imaging	databases	can	get	huge	most	save	images	permanently	store	baby
1	1	1	1	1	1	0	0	0	0	0	0
2	1	1	0	0	0	1	1	1	1	0	0
3	1	1	0	0	0	0	0	1	0	1	1
4	3	3	0	0	0	0	0	3	0	3	0

- Les valeurs du tableau sont toujours positives ou nulles. On se concentre sur les occurrences positives des termes.
- Les descripteurs (colonnes) sont exprimés dans les mêmes unités. Il n'y a pas de normalisation ou de standardisation des variables à effectuer.
- Certains termes sont présents dans peu de documents. Ou certains documents sont composés de peu de termes. De très nombreuses valeurs sont nulles.
- Mais les descripteurs sont issus des termes présents dans le corpus, une colonne composée exclusivement de la valeur 0 ne peut pas exister
- Il n'y a jamais de données manquantes (N/A). Un terme est forcément soit absent, soit présent dans un document.

La MDT n'est pas un tableau
de données comme les
autres.

MDT : Réduction de la dimensionnalité

Documents

Document	imaging	databases	can	get	huge	most	save	images	permanently	store	baby
1	1	1	1	1	1	0	0	0	0	0	0
2	1	1	0	0	0	1	1	1	1	0	0
3	1	1	0	0	0	0	0	1	0	1	1
4	3	3	0	0	0	0	0	3	0	3	0

Problème :

La dimensionnalité est souvent très élevée. Le nombre de colonnes peut excéder le nombre de lignes. Problème pour les algorithmes de data mining souvent (ex. calcul de la matrice variance covariance).

La réduction de la dimensionnalité est un enjeu crucial.

Pistes de solutions : Quelles pistes pour réduire la dimensionnalité (réduire la taille du dictionnaire) ?

- **Nettoyage** du corpus (ex. retrait des chiffres, correction orthographique, harmonisation de la casse, ...)
- Certains mots ne sont pas intrinsèquement porteurs de sens (ex. most) → **stopwords**
- Certains mots sont issus de la même forme canonique (**lemmatisation**) ou partagent la même racine (**stemming**) (ex. imaging vs. images)
- Certains mots sont synonymes ou recouvrent le même **concept** (ex. store vs. save)
- Certains mots apparaissent très souvent, d'autres très rarement (**filtrage sur les fréquences**)

MDT : Réduction de la dimensionnalité

- **Solution 1 : Définir le dictionnaire de manière ad hoc**

- Dans l'idéal, on connaît les prétraitements à réaliser, on connaît la liste des mots pertinents. Tout devient facile.

1. Imaging databases can get huge.
2. Most imaging databases save images permanently.
3. Imaging databases store images baby.
4. Imaging databases store images. Imaging databases store images. Imaging databases store images.



1. **image** databases can get huge
2. most **image** databases store **image** permanently
3. **image** databases store **image** baby
4. **image** databases store **image image** databases store **image image** databases store **image**

Document	databases	huge	image	permanently	store
1	1	1	1	0	0
2	1	0	2	1	1
3	1	0	2	0	1
4	3	0	6	0	3

Le nombre de 0 dans la table a été réduit de manière drastique.

Sauf situations très particulières, ce type de stratégie est impraticable sur des grands corpus.

MDT : Réduction de la dimensionnalité

- **Solution 2 : Le nettoyage du texte.**
- Le nettoyage dépend de ce que l'on souhaite exploiter par la suite.
- **La première passe de nettoyage est supprimer les stopwords :**
 - *stopwords* sont les mots très courants dans la langue étudiée qui **n'apportent pas de valeur informative** pour la compréhension du "sens" d'un document et corpus.
 - Il existe dans la librairie NLTK une liste par défaut des stopwords dans plusieurs langues, notamment le français et l'anglais.

Deuxième passe la lemmatisation ou la racinisation (stemming)

Le processus de « lemmatisation » consiste à représenter les mots (ou « lemmes ») sous leur forme canonique. Par exemple pour un verbe, ce sera son infinitif. Pour un nom, son masculin singulier. L'idée étant encore une fois de ne **conserver que le sens des mots** utilisés dans le corpus

MDT : Réduction de la dimensionnalité

- **Solution 2 : Le nettoyage du texte - Retrait des stopwords**

- Un mot vide est un mot communément utilisé dans une langue, non porteur de sens dans un document (ex. préposition, pronoms, etc.).
- Les mots vides ne permettent pas de distinguer les documents les uns des autres, ils sont inutilisables en text mining tel que nous le concevons (catégorisation, recherche d'information, etc.)
- **Exemple de mots vides en français :** avoir, bon, car, ce, cela, ces, ceux, chaque, ci, comme, comment, dans, des, du, dedans, dehors, depuis, devrait, doit, donc, dos, début, elle, elles, ...
- **Exemple de mots vides en anglais :** a, about, above, after, again, against, all, am,, an, and, any, are, aren't, as, cannot, could, couldn't, did, didn't, do, does, doesn't, doing, don't.....
- Listes complètes disponibles sur <https://www.ranks.nl/stopwords>



Nous pouvons compléter ou restreindre la liste des mots-clés en fonction du domaine d'étude ou du contexte dans lequel nous nous situons.

MDT : Réduction de la dimensionnalité

- **Solution 2 : Le nettoyage du texte - Retrait des stopwords**

Texte original

ohio mattress co said **its** first quarter ending february profits may **be below the** mln dlrs **or** cts a share earned **in the** first quarter **of** fiscal **the** company said **any** decline **would be** due **to** expenses related **to the** acquisitions **in the** middle **of the** current quarter **of** seven licensees **of** sealy inc **as well as** pct **of the** outstanding capital stock **of** sealy **because of these** acquisitions **it** said first quarter sales will **be** substantially higher **than** last years mln dlrs noting **that it** typically reports first quarter results **in** late march said **the** report **is** likely **to be** issued **in** early april **this** year **it** said **the** delay **is** due **to** administrative considerations including conducting appraisals **in** connection **with the** acquisitions reuter

125 termes

Texte après retrait des stopwords en anglais

ohio mattress co said first quarter ending february profits may mln dlrs cts share earned first quarter fiscal company said decline due expenses related acquisitions middle current quarter seven licensees sealy inc well pct outstanding capital stock sealy acquisitions said first quarter sales will substantially higher last years mln dlrs noting typically reports first quarter results late march said report likely issued early april year said delay due administrative considerations including conducting appraisals connection acquisitions reuter

76 termes



$\frac{125-76}{125} = 39.2\%$ des termes ont été retirés.

MDT : Réduction de la dimensionnalité

- **Solution 2 : Le nettoyage du texte - Retrait des stopwords -Remarques**

Les « stop word » sont en très grande partie composée de mots qui n'ont pas de sens en eux-mêmes, mais qui sont utilisés dans la construction des phrases (ex. prépositions, pronoms, verbes auxiliaires, articles). **Ils sont caractéristiques d'une langue et peuvent être utilisés pour les identifier.**

Algorithme

Entrée : document, liste des mots vides pour plusieurs langues

Sortie : la langue identifiée

Pour chaque langue :

Compter les occurrences des mots vides de la langue dans le document

La langue identifiée est celle pour laquelle le nombre de correspondance est le plus élevé



- Attention, si la taille de la liste des mots vides est différente d'une langue à l'autre, il faut en tenir compte.
- Il existe d'autres approches pour la détection de la langue, dont celles basées sur un apprentissage à partir de suites de caractères caractéristiques (n-grams).

MDT : Réduction de la dimensionnalité

- **Solution 2 : Le nettoyage du texte - Lemmatisation**

La lemmatisation consiste à analyser les termes de manière à identifier sa forme canonique (lemme), **qui existe réellement**. L'idée est de réduire les différentes formes (pluriel, féminin, conjugaison, etc.) en une seule.

Ex. am, are, is → be

car, cars, car's, cars' → car

Ainsi, la phrase « the boy's cars are different colors »
devient « the boy car be differ color ».



La technique fait à la fois référence à un dictionnaire, et à l'analyse morphosyntaxique des mots. Elle est spécifique à chaque langue. Des erreurs sont toujours possibles !

MDT : Réduction de la dimensionnalité

- **Solution 2 : Le nettoyage du texte - Stemming (Racinisation)**

Le stemming consiste à réduire un mot à sa racine (**stem**), **qui peut ne pas exister**.

L'algorithme de Porter est un des plus connus pour la langue anglaise. Il applique une succession de règles (mécaniques) pour réduire la longueur des mots c.-à-d. supprimer la fin des mots (voir [Wikipédia](#); Page de [Martin Porter](#)).

- Le stemming est un traitement final, qui n'autorise plus de post-traitements sur les mots
- Le stemming peut conduire à des regroupements erronés (ex. marmite, marmaille → marm)
- `nltk.stem` is a python package that performs stemming using different classes.
PorterStemmer is one of the classes

MDT : Réduction de la dimensionnalité

- **Solution 2 : Le nettoyage du texte - Lemmatisation**

ohio mattress co said first quarter ending february
profits may mln dlrs cts share earned first quarter fiscal
company said decline due expenses related acquisitions
middle current quarter seven licensees sealy inc well pct
outstanding capital stock sealy acquisitions said first
quarter sales will substantially higher last years mln dlrs
noting typically reports first quarter results late march
said report likely issued early april year said delay due
administrative considerations including conducting
appraisals connection acquisitions reuter

Avant stemming : 549 caractères

ohio mattress co said first quarter end februari
profit may mln dlrs cts share earn first quarter
fiscal compani said declin due expens relat
acquisit middl current quarter seven license seali
inc well pct outstand capit stock seali acquisit said
first quarter sale will substanti higher last year mln
dlrs note typic report first quarter result late
march said report like issu earli april year said delay
due administr consider includ conduct apprais
connect acquisit reuter

Après stemming : 477 caractères

MDT : Réduction de la dimensionnalité

- **Solution 2 : Le nettoyage du texte**

Sam Woods, a deputy governor of the Bank of England who has reviewed the contingency plans of more than 400 banks and financial firms, said he agreed with the findings of the survey.

preprocessing

sam, woods, governor, bank, england, contingency, plans, banks, firms, findings, survey

Le nettoyage peut comprendre l'harmonisation de la casse, le retrait des ponctuations et de \n, le retrait des nombres et le retrait des espaces en trop

MDT : Réduction de la dimensionnalité

- **Solution 3 : Utilisation d'un correcteur orthographique**
- Les coquilles peuvent fausser l'inventaire des termes. Utiliser un correcteur orthographique permet de réduire la dimensionnalité.
- Mode de fonctionnement d'un correcteur : L'outil s'appuie sur un dictionnaire où les termes sont correctement orthographiés.
 - Si le mot à évaluer y est présent → OK. S'il n'y est pas, on recense les mots les plus proches. Le ou les plus proches sont proposés.
- **Distance entre chaînes de caractères** : Une mesure spécifique aux chaînes de caractères doit être utilisée. La plus connue est celle de **Levenstein** (distance d'édition).
 - Attention de ne pas corriger à tort et à travers (ex. les noms propres)

MDT : Réduction de la dimensionnalité

- **Solution 3 : Utilisation d'un correcteur orthographique**
- **Distance de Levenshtein *** : On appelle distance de Levenshtein entre deux mots M et P le coût minimal pour aller de M à P en effectuant les opérations élémentaires suivantes :
 - substitution d'un caractère de M en un caractère de P ;
 - ajout dans M d'un caractère de P ;
 - suppression d'un caractère de M.
- On associe ainsi à chacune de ces opérations un coût. Le coût est toujours égal à 1, sauf dans le cas d'une substitution de caractères identiques.

* https://fr.wikipedia.org/wiki/Distance_de_Levenshtein

MDT : Réduction de la dimensionnalité

- Exemple d'utilisation de Distance de Levenshtein sous R :

```
> adist("totor", c("toto", "tata", "tutu", "tonton", "tantine"))  
      [,1] [,2] [,3] [,4] [,5]  
[1,]     1     3     3     2     5
```

La fonction renvoie un vecteur contenant la distance du mot à tester « totor » avec chaque mot du dictionnaire (« toto », « tata », ...).

- Exemple d'utilisation de la Distance de Levenshtein sous Python :

```
import Levenshtein as lev  
Distance = lev.distance("Benoit", "Ben")  
print("Levenshtein entre Benoit et Ben")  
print("> Distance: {0}".format(Distance))  
  
Levenshtein entre Benoit et Ben  
> Distance: 3
```

MDT : Réduction de la dimensionnalité

- **Solution 4 : Filtrage par la fréquence**

Fréquence : nombre de documents où le terme apparaît au moins une fois rapporté au nombre total de documents.

Document	databases	huge	image	permanently	store
1	1	1	1	0	0
2	1	0	2	1	1
3	1	0	2	0	1
4	3	0	6	0	3

- Fréquence trop élevée (termes présents dans presque tous les documents) : permet peut-être de cerner le domaine, mais ne permet pas de différencier les documents (ex. databases, image).
- Fréquence trop faible (termes présents dans de très rares documents) : ne permet pas de caractériser une différence significative entre les documents.

Le choix des seuils reste arbitraire.

PONDÉRATION

La Pondération

Plusieurs méthodes de pondération sont souvent utilisées en text Mining.

- **Pondération Binaire**
- **Fréquence des termes (TF : Term frequency) – ft,d**
- **Fréquences des termes – Différentes normalisations**
- **Inverse document frequency (IDF)**
- **Pondération TF-IDF**

La Pondération

- **Pondération Binaire :** Comptabiliser la présence de chaque terme dans le document, sans se préoccuper du nombre d'occurrences (de la répétition)

1. image databases can get huge
2. most image databases store image permanently
3. image databases store image baby
4. image databases store image image databases store image image databases store image



Document	databases	huge	image	permanently	store
1	1	1	1	0	0
2	1	0	1	1	1
3	1	0	1	0	1
4	1	0	1	0	1

Avantages

- Simplicité
- Forme de « lissage » de l'information en donnant la même importance à tous les termes
- Adaptée à certaines techniques (ex. règles d'association) et mesures de distance (ex. Jaccard)

Inconvénients

- Une partie de l'information n'est pas captée (perte d'information), dont pourrait tirer profit certaines catégories de techniques statistiques
- Pourquoi donner la même importance à tous les termes ?

La Pondération

- **Fréquence des termes (TF : Term frequency) – ft,d** : Pour un document, comptabiliser le nombre d'occurrence des termes. **Indicateur de l'importance du terme dans le document.**

1. image databases can get huge
 2. most image databases store image permanently
 3. image databases store image baby
 4. image databases store image image databases store image image databases store image



Document	databases	huge	image	permanently	store
1	1	1	1	0	0
2	1	0	2	1	1
3	1	0	2	0	1
4	3	0	6	0	3

Avantages

- On capte plus d'information, la répétition d'un terme dans le document est prise en compte
- Des techniques savent prendre en compte ce type d'information (calcul matriciel)

Inconvénients

- Les écarts entre documents sont exagérés (ex. si on utilise une distance euclidienne)
- On ne tient pas compte de la prévalence des termes dans l'ensemble des documents (cf. IDF)

La Pondération

- **Fréquences des termes – Différentes normalisations**

$f_{t,d}$	Document	databases	huge	image	permanently	store
	1	1	1	1	0	0
	2	1	0	2	1	1
	3	1	0	2	0	1
	4	3	0	6	0	3

On simplifie en considérant que seuls ces termes sont apparus.

Les normalisations des fréquences permettent d'amortir les écarts et/ou de tenir compte de la longueur des documents.

1. Normalisation logarithmique
2. Double normalisation 0.5
3. Normalisation simple

La Pondération


Fréquences des termes – Différentes normalisations

Normalisation logarithmique

$$tf(t, d) = \begin{cases} 0 & \text{si } f_{t,d} = 0 \\ 1 + \log_{10} f_{t,d} & \text{sinon} \end{cases}$$


Document	databases	huge	image	permanently	store
1	1.00	1.00	1.00	0.00	0.00
2	1.00	0.00	1.30	1.00	1.00
3	1.00	0.00	1.30	0.00	1.00
4	1.48	0.00	1.78	0.00	1.48


Double normalisation 0.5

$$tf(t, d) = 0.5 + 0.5 \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$$


Document	databases	huge	image	permanently	store
1	1.00	1.00	1.00	0.50	0.50
2	0.75	0.50	1.00	0.75	0.75
3	0.75	0.50	1.00	0.50	0.75
4	0.75	0.50	1.00	0.50	0.75

On pondère la fréquence par le nombre d'apparition du terme le plus fréquent du document (une manière de tenir compte de la longueur)

Normalisation simple

$$tf(t, d) = \frac{f_{t,d}}{\sum_{t'} f_{t',d}}$$


Document	databases	huge	image	permanently	store
1	0.33	0.33	0.33	0.00	0.00
2	0.20	0.00	0.40	0.20	0.20
3	0.25	0.00	0.50	0.00	0.25
4	0.25	0.00	0.50	0.00	0.25

On pondère la fréquence par le nombre de termes présents dans le document (une autre manière de tenir compte de la longueur)

La Pondération

Inverse document frequency (IDF)

Un terme présent dans presque tout le corpus (D) influe peu quand il apparaît dans un document. A l'inverse, un terme rare apparaissant dans un document doit retenir notre attention. **L'IDF mesure l'importance d'un terme dans un corpus.**

$$idf(t,D) = \log_{10} \frac{N}{n_t}$$

N : nombre de documents dans le corpus t

n : nombre de documents où le terme apparaît

Document	databases	huge	image	permanently	store
1	1	1	1	0	0
2	1	0	2	1	1
3	1	0	2	0	1
4	3	0	6	0	3
n_t	4	1	4	1	3
idf(t,D)	0.000	0.602	0.000	0.602	0.125

« databases » et « image » sont des termes « évidents » qui ne peuvent pas servir à la différenciations des documents.

Le filtrage sur les fréquences permet d'éviter l'influence exagérée des termes très rares.

Remarque : Une formule lissée est utilisée parfois pour éviter d'obtenir un **IDF nul**

$$idf(t,D) = \log_{10} (1 + \frac{N}{n_t})$$

La Pondération

- **Pondération TF-IDF** : Relativiser l'importance d'un terme dans un document (**TF**) par son importance dans le corpus (**IDF**).

Dans sa forme la plus utilisée
(mais des variantes sont possibles)

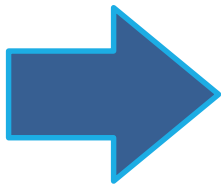
$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \times \text{idf}(t, D)$$

$$\text{tfidf}(t, d, D) = f_{t,d} \times \log_{10} \frac{N}{n_t}$$

Le terme apparaît beaucoup
dans le document.

Il se fait rare par ailleurs.

Le TF-IDF d'un terme dans
un document est élevé
quand :



Document	databases	huge	image	permanently	store
1	0.00	0.60	0.00	0.00	0.00
2	0.00	0.00	0.00	0.60	0.12
3	0.00	0.00	0.00	0.00	0.12
4	0.00	0.00	0.00	0.00	0.37

Basics of CountVectorizer

Basics of CountVectorizer

- Countvectorizer is a method to convert text to numerical data.
- Countvectorizer makes it easy for text data to be used directly in machine learning and deep learning models such as text classification.
- To show you how it works let's take an example:
corpus = ['This is the first document.',
 'This is the second document.',
 'And this is the third one.',
 'Is this the first document?']

Basics of CountVectorizer

```
from sklearn.feature_extraction.text import CountVectorizer
```

```
# Sample corpus of documents
```

```
corpus = ['This is the first document.',  
          'This is the second document.',  
          'And this is the third one.',  
          'Is this the first document?']
```

the unique words in the vocabulary:

```
['and' 'document' 'first' 'is' 'one' 'second' 'the' 'third' 'this']
```

```
# Create an instance of CountVectorizer
```

```
vectorizer = CountVectorizer()
```

```
# Fit and transform the corpus of documents
```

```
count_matrix = vectorizer.fit_transform(corpus)
```

```
# Print the feature names (i.e., the unique words)
```

```
print('the unique words in the vocabulary:')
```

```
print(vectorizer.get_feature_names_out())
```

```
# Print the matrix of token counts
```

```
count_array = count_matrix.toarray()
```

```
df = pd.DataFrame(data=count_array, columns = vectorizer.get_feature_names_out())
```

```
print('dataframe representation:')
```

```
df|
```

	and	document	first	is	one	second	the	third	this
0	0	1	1	1	0	0	1	0	1
1	0	1	0	1	0	1	1	0	1
2	1	0	0	1	1	0	1	1	1
3	0	1	1	1	0	0	1	0	1

Basics of CountVectorizer

vectorizer.vocabulary_ est un dictionnaire qui mappe chaque mot unique dans le vocabulaire généré par CountVectorizer à un index entier. Cet index correspond à la colonne correspondante dans la matrice d'occurrence de termes retournée par transform().

```
vectorizer.vocabulary_
```

```
{'this': 8,  
 'is': 3,  
 'the': 6,  
 'first': 2,  
 'document': 1,  
 'second': 5,  
 'and': 0,  
 'third': 7,  
 'one': 4}
```

Basics of CountVectorizer

Sample corpus of documents

```
corpus = ['This is the first document.',  
          'This is the second document. And this is the third one.',  
          'Is this the first document?']
```

Create an instance of CountVectorizer

```
vectorizer = CountVectorizer()
```

Fit and transform the corpus of documents

```
count_matrix = vectorizer.fit_transform(corpus)
```

Print the feature names (i.e., the unique words

```
print('the unique words in the vocabulary:')
```

```
print(vectorizer.get_feature_names_out())
```

Print the matrix of token counts

```
count_array = count_matrix.toarray()
```

```
df = pd.DataFrame(data=count_array, columns = vectorizer.get_feature_names_out())
```

```
print('dataframe representation:')
```

```
df
```

the unique words in the vocabulary:

['and' 'document' 'first' 'is' 'one' 'second' 'the' 'third' 'this']

dataframe representation:

	and	document	first	is	one	second	the	third	this
0	0	1	1	1	0	0	1	0	1
1	1	1	0	2	1	1	2	1	2
2	0	1	1	1	0	0	1	0	1

Basics of CountVectorizer

- There Are Some Additional Parameters In The Count Vector.
 - Stop Words
 - Min_df
 - Max_df
 - Binary

Basics of CountVectorizer- Stop Words

```
# Sample corpus of documents
corpus = ['This is the first document.',
          'This is the second document.',
          'And this is the third one.',
          'Is this the first document?']

# Create an instance of CountVectorizer with options
vectorizer = CountVectorizer(stop_words='english', lowercase=True)

# Fit and transform the corpus of documents
count_matrix = vectorizer.fit_transform(corpus)

# Print the feature names (i.e., the unique words in the vocabulary)
print('the unique words in the vocabulary:')
print(vectorizer.get_feature_names_out())

# Print the matrix of token counts
count_array = count_matrix.toarray()
df = pd.DataFrame(data=count_array, columns = vectorizer.get_feature_names_out())
print('dataframe representation:')
df
```

the unique words in the vocabulary:
['document' 'second']
dataframe representation:

	document	second
0	1	0
1	1	1
2	0	0
3	1	0

Basics of CountVectorizer: Min_df& Max_df

min_df et max_df sont des options dans CountVectorizer qui permettent de contrôler le vocabulaire en fixant des seuils de fréquence documentaire pour les mots.

min_df spécifie la fréquence minimale avec laquelle un mot doit apparaître dans les documents pour être inclus dans le vocabulaire.

- Par défaut, min_df est défini à 1, ce qui signifie que tous les mots qui apparaissent dans au moins un document sont inclus dans le vocabulaire.

max_df spécifie la fréquence maximale avec laquelle un mot peut apparaître dans les documents pour être inclus dans le vocabulaire.

- max_df peut être défini soit en tant que fraction de la taille du corpus (max_df=0.5 signifie que les mots qui apparaissent dans plus de la moitié des documents sont ignorés), soit en tant que nombre entier (max_df=2 signifie que les mots qui apparaissent dans plus de deux documents sont ignorés).

Basics of CountVectorizer : Min_df& Max_df

```
# Sample corpus of documents
corpus = ['This is the first document.',
          'This is the second document.',
          'And this is the third one.',
          'Is this the first document?']

# Create an instance of CountVectorizer with options
vectorizer = CountVectorizer(stop_words='english', lowercase=True, min_df=2, max_df=0.75)

# Fit and transform the corpus of documents
count_matrix = vectorizer.fit_transform(corpus)

# Print the feature names (i.e., the unique words in the vocabulary)
print('the unique words in the vocabulary:')
print(vectorizer.get_feature_names_out())

# Print the matrix of token counts
count_array = count_matrix.toarray()
df = pd.DataFrame(data=count_array, columns = vectorizer.get_feature_names_out())
print('dataframe representation:')
df
```

the unique words in the vocabulary:
['document']
dataframe representation:

document	
0	1
1	1
2	0
3	1

Basics of CountVectorizer : Binary

```
# Sample corpus of documents
corpus = ['This is the first document.',
          'This is the second document.',
          'And this is the third one.',
          'Is this the first document?']

# Create an instance of CountVectorizer with options
vectorizer = CountVectorizer(binary=True)

# Fit and transform the corpus of documents
count_matrix = vectorizer.fit_transform(corpus)

# Print the feature names (i.e., the unique words in the vocabulary)
print('the unique words in the vocabulary:')
print(vectorizer.get_feature_names_out())

# Print the matrix of token counts
count_array = count_matrix.toarray()
df = pd.DataFrame(data=count_array, columns = vectorizer.get_feature_names_out())
print('dataframe representation:')
df
```

the unique words in the vocabulary:
['and' 'document' 'first' 'is' 'one' 'second' 'the' 'third' 'this']
dataframe representation:

	and	document	first	is	one	second	the	third	this
0	0	1	1	1	0	0	1	0	1
1	0	1	0	1	0	1	1	0	1
2	1	0	0	1	1	0	1	1	1
3	0	1	1	1	0	0	1	0	1

Basics of CountVectorizer : vectorizer.transform

```
from sklearn.feature_extraction.text import CountVectorizer

# Define the corpus of documents
corpus = ['This is the first document.',
          'This is the second document.',
          'And this is the third one.',
          'Is this the first document?']

# Create an instance of CountVectorizer and fit on the corpus
vectorizer = CountVectorizer()
vectorizer.fit(corpus)

# New sentence to be vectorized
new_sentence = 'This is the first document, but it is not the last one.'

# Transform the new sentence into a vector
new_vector = vectorizer.transform([new_sentence])

# Print the feature names (i.e., the unique words in the vocabulary)
print(vectorizer.get_feature_names_out())

# Print the vector representation of the new sentence
print(new_vector.toarray())
```