

Disclaimer

Bonjour, vous êtes dans ce document Word un peu comme dans ma tête. Il se peut que vous repartiez avec plus de questions que de réponse. Mais ça peut aussi très bien se passer. J'ai essayé d'organiser le document afin qu'il soit plus lisible aux hommes du dehors (comme vous).

Table des matières

Disclaimer	1
Analyse de l'article donnée	1
Introduction de l'article.....	2
Calcul de la longueur d'un topic.....	6
Evolution journalière	6
01/06/2022.....	6
03/06/2022.....	7
07/06/2022.....	10
08/06/2022.....	12
09/06/2022.....	15
10/06/2022.....	23

Analyse de l'article donnée

Page article : <https://arxiv.org/abs/2010.01600>

On utilise un modèle de « topic » (topic modeling) qui compare des topics qu'on identifie comme soit :

- Long-lasting trends
- Short-lasting topic

Le modèle doit aussi savoir localiser ses topics dans le temps avec clarté.

Cet article présente différents modèles.

On compare la « variabilité » (variability) des tailles/longueur de divers topics.

Ces longueurs étant celles obtenues en utilisant différents modèles (de topic) assez connus :

- Latent Dirichlet allocation (LDA)
- Nonnegative matrix factorization (NMF) (sujet de mon stage)
- Nonnegative CANDECOMP/PARAFAC tensor decomposition (NCPD et Online NCPD)

Le NCPD est censé être la « tensor counterpart » de la NFM. Cela veut sûrement dire que c'est des modèles similaires mais que le NCPD utilise les tensor contrairement au NFM. (?)

L'article veut nous montrer que seules les méthodes utilisant des tensor arrivent à détecter les short-lasting topics (topics de courte de durée) avec succès.

De plus, les méthodes utilisant les tensors sont plus précises (accurate) pour connaître le moment d'apparition et de disparition d'un topic. Elles sont plus précises que la LDA et le NMF.

L'article propose de faire des mesures « quantitatives » de la taille des topics. Cela servira à illustrer la performance du NCPD lorsqu'il faut trouver/découvrir des topics. Les topics utilisés seront pour certains « semi-synthetic » ou issues de données réels (real-world data). Les données comprennent des intitulés de « news » (news headlines) ainsi que des tweets liés au covid-19.

Les données « semi-synthetic » sont sûrement des données en partie inventées. Peut être à partir d'autre données réelles ?

Introduction de l'article

Quel est le format des données ?

Modèle LDA :

- Modélise les topics ou les classe par probability de distribution de « set de mots » (set of words).
- Ces sets de mots évoluent selon un « scheme Bayésien » (Bayesian scheme) en donnant au « batches » une matrice « words x documents ». C'est sûrement un tableau contenant le nombre de fois qu'un mot apparaît pour chaque document.
- Il semble qu'on reçoit en sortie deux « représentations » de type « words x topics » et « topics x documents ». Cela veut sûrement dire qu'on a en sortie deux autres matrices créées par notre modèle. Le modèle a dû créer lui-même les « topics » en analysant les mots par documents. Puis il a dû faire des associations et compter le nombre de topics par document ainsi que la relation entre un mot et un topic identifié.

Modèle NMF :

- C'est aussi un modèle qui utilise des matrices (comme le LDA). Il va prendre la matrice « words x documents » et la « décomposer » en « words x topics » et « topics x documents »
- Les documents peuvent avoir un « timestamp ». C'est sûrement un fichier ou un marquage qui permet d'indiquer le moment où est publié le texte. Si le « timestamp » est ordonné en fonction du temps, la matrice « topics x document » pourra trier et présenter les topics dans cet ordre.

Il y a deux façons de faire que le NMF prenne en compte les « tranches » (slices) temporelles simultanées ou simultanément

- On peut factoriser chaque tranche temporelle

“There are two basic methods of using NMF for dynamic topic modeling in a way that accounts to the time slices that are essentially simultaneous for the considered dynamic. First, one can factorize each time slice independently using NMF and obtain topic structure for each time slice [35, 38, 3, 36]. Second, one can concatenate all the time slices along the documents dimension and decompose the resulting matrix using NMF with a fixed dictionary to obtain common topics. The evolution of such topics is then found by computing their contribution in each time slice [18]. In this work, we do the latter to post-process both NMF and LDA for better visualization and more interpretable results on the time scale. We discuss the methods in more details in Section 2. »

Concatenate all the time slices''

On peut encoder le corpus de documents dans un tensor à 3 dimensions

Qu'est-ce que le entrywise Froebenius norm ?

unigrams and bigrams

2.1 Nonnegative Matrix Factorization

Nonnegative Matrix Factorization (NMF) is a popular tool for extracting hidden themes from text data [13, 32]. For a data matrix $\mathbf{X} \in \mathbb{R}_+^{m \times n}$, one learns a low-rank dictionary $\mathbf{W} \in \mathbb{R}_+^{m \times r}$ and code matrix $\mathbf{H} \in \mathbb{R}_+^{r \times n}$ that minimize $\|\mathbf{X} - \mathbf{WH}\|_F^2$, where $r > 0$ is typically chosen such that $r < \min\{m, n\}$. Suppose m denotes the number of features (in our case unigrams and bigrams) and n the number of documents, then the dictionary matrix \mathbf{W} represents *topics* in terms of the original features. Each column of the code matrix \mathbf{H} represents a data point as a linear combination of the dictionary elements with nonnegative coefficients. We use NMF to learn a dictionary \mathbf{W} from all data and analyze topic dynamics through changes in topic prevalence over time in the code matrices \mathbf{H}_i from each time slice i . The matrices \mathbf{H}_i are column blocks of the code matrix \mathbf{H} that correspond to all the documents from the i -th time slice. The mean topic representation for each time slice is then calculated and given in the columns of the heatmaps (e.g. Figure 8).

Comment fait l'algorithme pour extraire des « hidden themes » d'un texte ?

Il devrait pouvoir :

- Savoir ce qu'est un themes
- Savoir le trouver dans un texte
- Savoir l'extraire

Qu'est-ce qu'un « low rank dictionary \mathbf{W} » ainsi qu'un « code matrix \mathbf{H} » ?

- Comment obtient-on ces deux matrices à partir de la première ?
- Quelle sorte d'information contiennent ces deux nouvelles matrices ?

\mathbf{W} contient les topics qui ont été repérés par notre modèle dans une étape précédente. \mathbf{W} est de dimension $m \times r$. « m » étant le nombre de « features ». « r » étant choisi de sorte à être plus petit que m (nombre de features) et n .

- Qu'est-ce qu'un « feature » ? On nous dit qu'ils sont ici sous forme de « unigrams and bigrams ». On nous dit aussi que \mathbf{W} représente les topics sous forme de « original features ».

features nom, pluriel ʁʒ (singulier: feature)

caractéristiques pl f ʁʒ

The new software has some useful features.

Le nouveau logiciel possède des caractéristiques utiles.

plus rare :

éléments pl m ʁʒ · traits pl m ʁʒ · dispositifs pl m ʁʒ · fonctions pl f ʁʒ · longs métrages pl m ʁʒ · particularités pl

options pl f ʁʒ · composantes pl f ʁʒ · aspects pl m ʁʒ · articles pl m ʁʒ

-

This is Big Data AI Book						
Uni-Gram	This	is	Big	Data	AI	Book
Bi-Gram	This is	is Big	Big Data	Data AI	AI Book	
Tri-Gram	This is Big	is Big Data	Big Data AI	Data AI Book		

-
- J'en déduis que notre modèle après avoir trouvé les topics leur a donné des noms « features » composés d'un ou deux mots.

Anglais (langue détectée) ▼

W represents topics in terms of the original features

↔

Français ▼

W représente les sujets en termes de caractéristiques originales

- fonctionnalités ...
- fonctions ...
- traits ...
- dispositifs ...
- particularités ...

-
-
- $W_{i,j}$ représente le nombre de fois qu'un mot est utilisé dans un topic i apparaît en fonction de son rang j . Soit $W_{i,j}$ contient un feature. Le nombre de mot r étant
- Il semble que l'ensemble de ces matrices doit vérifier une condition, ils doivent minimiser la distance au carré entre X et WH . Je ne suis pas
- W associe à chaque topic un feature (unigram ou bigram), chaque $W_{i,j}$ contient le nombre de fois que cette association est faite.
- Chaque colonne de H représente une donnée comme une combinaison linéaire d'éléments du « dictionary ».

Qu'est-ce que sont les unigrams et bigrams ?

"one learns a low-rank dictionary W " – on fait apprendre cette matrice à notre modèle ?

NMF Algorithm Basics

An ML algorithm that, given a matrix V of dimension $f \times t$, "learns" V 's approx factors: W of dimension $f \times k$ and H of dimension $k \times t$, where $k < \min(f, t)$

- All matrices must contain only non-negative values
- W and H can be multiplied together to make V' , an approx. to V

$$\begin{matrix} W \\ \begin{bmatrix} \square & \square \\ \square & \square \\ \square & \square \\ \square & \square \end{bmatrix} \end{matrix} \times \begin{matrix} H \\ \begin{bmatrix} \square & \square & \square & \square & \square & \square \\ \square & \square & \square & \square & \square & \square \end{bmatrix} \end{matrix} \approx \begin{matrix} V \\ \begin{bmatrix} \square & \square & \square & \square & \square & \square \\ \square & \square & \square & \square & \square & \square \\ \square & \square & \square & \square & \square & \square \\ \square & \square & \square & \square & \square & \square \end{bmatrix} \end{matrix}$$

Regarder en mode

An NMF Implementation - Derivation

To begin, we frame an optimization problem

- **Kullback-Leibler:**

$$D(\mathbf{V} \parallel \hat{\mathbf{V}}) = \sum_{i,j} \left(\mathbf{V}_{ij} \log \frac{\mathbf{V}_{ij}}{\hat{\mathbf{V}}_{ij}} - \mathbf{V}_{ij} + \hat{\mathbf{V}}_{ij} \right)$$

Minimize this function, by using Majorization-Minimization and applying Jensen's Inequality ...

Les normes classiques sur un espace de dimension $n \times p$ sont :

$$\|M\|_1 = \sum_{i=1}^n \sum_{j=1}^p |M_{i,j}|$$

$$\|M\|_2 = \sqrt{\sum_{i=1}^n \sum_{j=1}^p |M_{i,j}|^2}$$

et plus généralement :

$$\|M\|_q = \left(\sum_{i=1}^n \sum_{j=1}^p |M_{i,j}|^q \right)^{\frac{1}{q}}$$

pour $q \in [1; +\infty[$
mais aussi :

$$\|M\|_\infty = \max_{i=1}^n (\max_{j=1}^p |M_{i,j}|)$$

<https://www.youtube.com/watch?v= QiTQQDrx5I>

<https://www.youtube.com/watch?v=ZspR5PZemcs>

Calcul de la longueur d'un topic

Our proposed metric quantifies the number of consecutive days required to cover a certain “proportion” of the topic that we denote by α . We consider the matrix $\tilde{\mathbf{T}} \in \mathbb{R}_+^{r \times n}$ which is the matrix \mathbf{T} with the rows normalized to add up to 1. Normalization of the rows produces a probability distribution for each individual topic over time. Informally, it captures how many consecutive days are required for each topic to include a certain proportion of its whole “mass”. Specifically, for a fraction $\alpha \in [0, 1]$ and the topic τ , its α -effective length denoted by $\ell_\alpha(\tau)$, is defined as

$$\ell_\alpha(\tau) := \max_{i \in [n]} \begin{cases} \min \{l | \sum_{i:i+l} > \alpha\} & \text{if } \sum_{i:n} > \alpha, \\ 0 & \text{otherwise,} \end{cases} \quad (3.1)$$

where $\sum_{i_1:i_2} := \sum_{j=i_1}^{i_2} \tilde{\mathbf{T}}[\tau, j]$.

Evolution journalière

01/06/2022

Recherchons dans le document des informations sur comment se fait la décomposition de la matrice initiale en deux matrices.

Rappelons qu'il faut faire plusieurs fois cette opération.

La recherche dans l'article n'a pas été fructueuse. On nous dit qu'on crée les matrices W et H mais je ne sais pas exactement comment.

Regardons plutôt les ressources git à disposition :

- Le repo où déposer nos fichiers
- Un repo git où se trouvent le travail de Hanbaek lyu et Lara kassab sur le topic modelling
- Ainsi qu'un « notebook » qui contient un code de programme pour faire du NMF, LDA, NCPD et ONCPD.

J'ai décidé d'exécuter le code de Lyu et Lara dans un premier temps.

J'ai tout d'abord effectué une copie du repo qu'ils proposent en local puis il a fallu ajouter les données « ABC news ».

Ouvrir jupyter notebook et suivre les consignes du README.

Recreating experiments

Files for reproducing experimental results can be found in the `scripts` folder.

- The notebook `methods_semisynthetic_20news.ipynb` reproduces results for the semi-synthetic 20 news dataset.
- The notebook `methods_headlines_dataset.ipynb`, reproduces results for the news headlines dataset. This notebook will likely take a while (hours) to run.
- The python file `produce_twitter_figures.py` reproduces results on COVID19 related tweets. In accordance with Twitter's policies, we do not provide data for this experiment. Tweet IDs were accessed from <https://github.com/echen102/COVID-19-TweetIDs>.
- The python file `OCPDL_benchmark.py` reproduces the reconstruction error comparison plots for NCPD and Online NCPD.

Cette partie du readme mis à disposition va nous faciliter la tâche afin de tout exécuter. Nous pouvons ouvrir chacun de ces notebooks afin d'essayer une ou plusieurs méthodes de topic modeling sur des données différentes.

Voyons comment ça se passe

On décide d'exécuter le notebook « `methods_semisynthetic_20news.ipynb` »

Voyons ce qu'il fait par étape

```
Load 20newsgroups dataset

Entrée [ ]: # Download the data with the selected categories
remove_tuple = ('headers', 'footers', 'quotes') # remove headers, footers, and quotes from all documents
newsgroups_subset_atheism = fetch_20newsgroups(subset='all', categories=['alt.atheism'], shuffle=True, random_state = 1,
remove=remove_tuple)
newsgroups_subset_space = fetch_20newsgroups(subset='all', categories=['sci.space'], shuffle=True, random_state = 1,
remove=remove_tuple)
newsgroups_subset_forsale = fetch_20newsgroups(subset='all', categories=['misc.forsale'], shuffle=True, random_state = 1,
remove=remove_tuple)
newsgroups_subset_baseball = fetch_20newsgroups(subset='all', categories=['rec.sport.baseball'], shuffle=True, random_state = 1,
remove=remove_tuple)
newsgroups_subset_windowsx = fetch_20newsgroups(subset='all', categories=['comp.windows.x'], shuffle=True, random_state = 1,
remove=remove_tuple)
```

Il charge les données mais ne les fait pas d'un coup, on dirait qu'il y a une sorte de découpage et de réorganisation.

Ah on dirait qu'il ne charge pas tout le dataset mais uniquement certaines colonnes. Mais je ne sais pas en détail ce qu'il fait.

03/06/2022

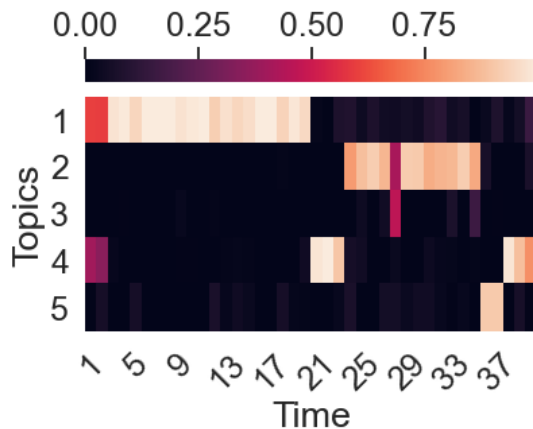
Nous avons pu continuer avec le même code et lancer une NMF, NCPD et LDA



```

11 save_figures:
    plt.savefig(os.path.join(local_path, "NCPD_Normalized_Topic_Time_20news.png"), bb
plt.show()

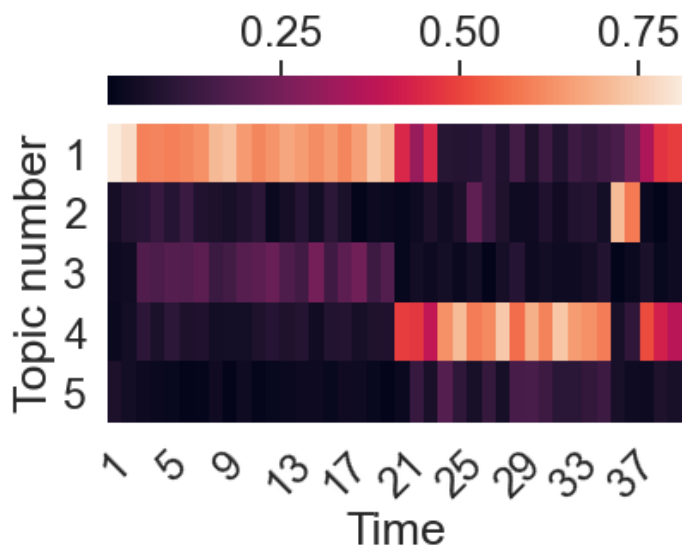
```



```

plt.savefig(os.path.join(local_path, "LDA_Normalized_Topic_Time_20news.png"), bb
plt.show()

```



Mystères :

- La façon de découper et réorganiser les données en matrices et tensors
- La façon de choisir les topics (exemple LDA)

```

Topic 1: would, like, one, space, think
Topic 2: edu, use, window, system, subject
Topic 3: space, launch, nasa, shuttle, mission
Topic 4: new, sale, please, one, offer
Topic 5: 00, 50, 20, 10, 40

```

- La structure de départ des données

En résumé le code a eu un peu de mal à s'exécuter. Il y avait des bibliothèques à installer ainsi que quelques variables dont les noms devaient être un peu modifiées.

Essayons d'exécuter l'ONCPD en analysant de plus près les parties ou alors en montrant ce que je ne comprends pas

Les étapes de l'ONCPD dans le programme :

- # SVD initialization (Optional)
- # Run Online NCPD
- # Obtain the shape of the factor matrices
- # Display topics
- # Visualize topic distributions

J'ai tenté de trouver une ressource expliquant la nmf et je suis tombé sur ce site :

<https://towardsdatascience.com/nmf-a-visual-explainer-and-python-implementation-7ecdd73491f8>

C'est beau. C'est de : Anupama Garla



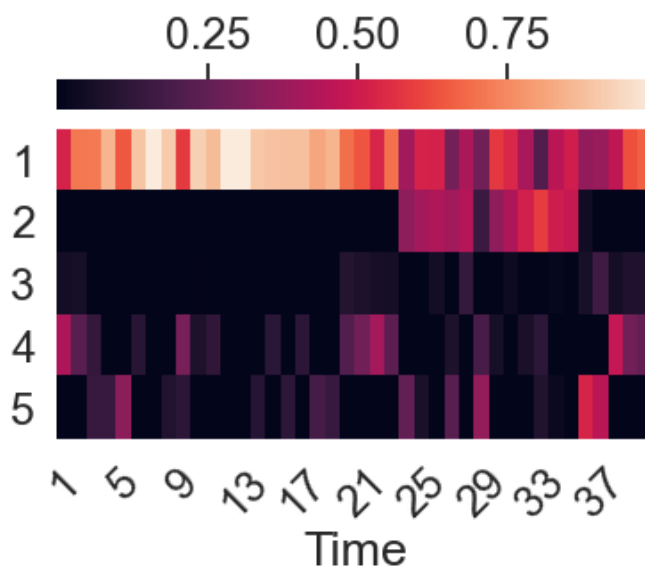
Anupama Garla

117 Followers

A Data Scientist who wants to help make beautiful meaningful insight-based actions. Let's connect on LinkedIn! <https://www.linkedin.com/in/anupama-garla/>

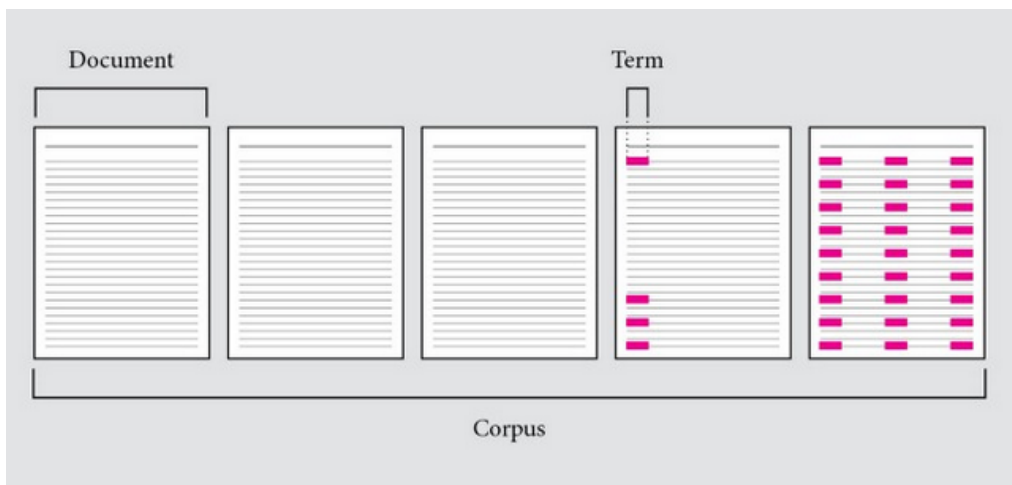
Sinon voici ce que me rend l' ONCPD

```
if save_figures:
    plt.savefig(os.path.join(local_path, "ONCPD_Normalized_Topic_Time_20news.png"),
    plt.show())
```



07/06/2022

Je suis de retour et nous allons lire l'article de anupama. Aujourd'hui j'ai malheureusement oublié ma trousse chez moi, ça ne va pas du tout. Passons.



‘This hypothesis is often stated in terms like “words which are similar in meaning occur in similar contexts” (Rubenstein & Goodenough, 1965)’

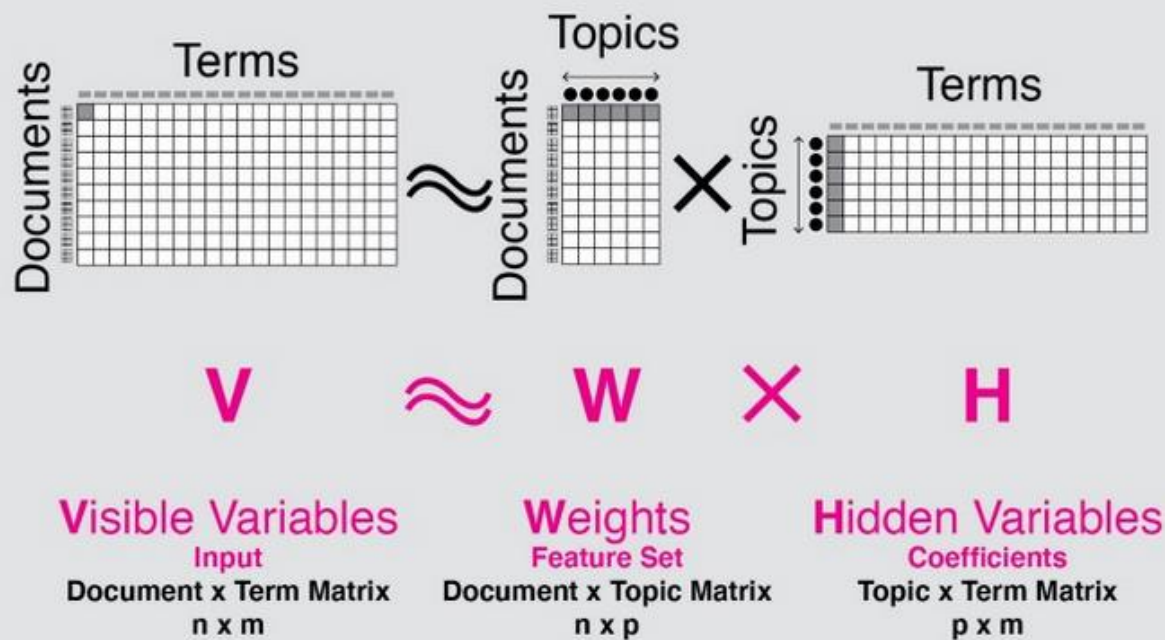
and continues — ‘The general idea behind the distributional hypothesis seems clear enough: there is a correlation between distributional similarity and meaning similarity, which allows us to utilize the former in order to estimate the latter.’

Mais c’est trop bien cette théorie.

On peut donc détecter un topic à l’aide de la distribution des termes le long d’un document ?

For us, this means that documents can be treated as a bag of words. Similar documents have similar word frequency distributions — and similar topics. Two words that occur together are likely similar — belonging to a single topic. This algorithm ignores *syntactic information* aka word order, and *semantic information* aka multiple meanings.

Non-Negative Matrix Factorization Generic Diagram



Le retour de la mystérieuse décomposition. Ce qui est mystérieux est la méthode exacte de « décomposition ». Puis comment trouve-t-on les topics ?

J'imaginai jusque là qu'une sorte d'analyse permet de les trouver.

Ce que je sais, c'est qu'il faut en trouver r .

visible variables. For image processing, the visible variables would be specific pixels. Here is an excerpt of a *V Matrix* of Presidential Inauguration Speech Text:

	constitution	union	peace	freedom	principle	man	spirit	party	congress	justice	year	day	policy	president	service
Name															
Theodore Roosevelt	0.00	0.00	0.10	0.00	0.00	0.00	0.12	0.00	0.00	0.07	0.00	0.10	0.00	0.00	0.00
William Howard Taft	0.02	0.00	0.03	0.01	0.02	0.02	0.00	0.02	0.13	0.02	0.04	0.01	0.12	0.01	0.00
Woodrow Wilson	0.00	0.00	0.00	0.00	0.03	0.05	0.03	0.14	0.00	0.17	0.07	0.09	0.03	0.06	0.03
Warren G. Harding	0.00	0.02	0.09	0.07	0.00	0.03	0.05	0.01	0.02	0.06	0.00	0.00	0.05	0.00	0.08
Calvin Coolidge	0.07	0.00	0.15	0.09	0.08	0.05	0.00	0.17	0.03	0.09	0.04	0.01	0.10	0.02	0.03
Herbert Hoover	0.02	0.00	0.16	0.07	0.00	0.03	0.02	0.08	0.04	0.15	0.03	0.04	0.04	0.00	0.09
Franklin D. Roosevelt	0.02	0.00	0.00	0.00	0.00	0.00	0.06	0.00	0.10	0.00	0.02	0.08	0.06	0.00	0.00
Harry S. Truman	0.00	0.00	0.20	0.21	0.05	0.07	0.00	0.00	0.00	0.09	0.03	0.00	0.02	0.02	0.00
Dwight D. Eisenhower	0.02	0.00	0.19	0.18	0.10	0.11	0.02	0.00	0.00	0.00	0.00	0.05	0.02	0.00	0.03
John F. Kennedy	0.00	0.00	0.10	0.12	0.00	0.12	0.00	0.03	0.00	0.03	0.05	0.05	0.00	0.12	0.03
Lyndon Baines Johnson	0.00	0.24	0.00	0.05	0.00	0.25	0.02	0.00	0.00	0.13	0.06	0.08	0.00	0.00	0.00
Richard Milhous Nixon	0.02	0.00	0.22	0.02	0.00	0.21	0.12	0.00	0.00	0.02	0.11	0.04	0.00	0.07	0.00
Jimmy Carter	0.00	0.00	0.03	0.13	0.06	0.03	0.19	0.00	0.00	0.03	0.06	0.03	0.03	0.10	0.00
Ronald Reagan	0.02	0.00	0.03	0.15	0.02	0.11	0.00	0.02	0.02	0.02	0.05	0.10	0.00	0.10	0.00
George Bush	0.00	0.02	0.05	0.09	0.02	0.10	0.00	0.02	0.07	0.03	0.03	0.15	0.00	0.12	0.00
Bill Clinton	0.00	0.00	0.00	0.05	0.00	0.00	0.02	0.00	0.06	0.00	0.00	0.00	0.00	0.05	0.10
George W. Bush	0.00	0.02	0.02	0.08	0.06	0.00	0.06	0.00	0.00	0.06	0.04	0.04	0.00	0.07	0.06
Barack Obama	0.00	0.00	0.07	0.06	0.02	0.06	0.10	0.00	0.00	0.00	0.05	0.09	0.00	0.02	0.06
Donald J. Trump	0.00	0.00	0.00	0.03	0.00	0.00	0.03	0.05	0.00	0.02	0.05	0.09	0.00	0.14	0.00

Excerpt of V Matrix (Using TF-IDF Vectorizer) by Anupama Garla

Voilà une chose surprenante. On a pour la première fois un exemple visuel de matrice qu'on prend en entrée pour la NMF. Cependant on y retrouve des nombres à virgule. Des proportions ?

On veut ensuite nous montrer la matrice W.

	Union + Constitution	Man + Freedom	Business + Policy	Principles + Improvement	Family + Jobs
Name					
George Washington	0.00	0.00	0.00	0.55	0.01
John Adams	0.03	0.00	0.02	0.53	0.03
Thomas Jefferson	0.05	0.19	0.00	0.37	0.00
James Madison	0.00	0.00	0.00	0.61	0.00
James Monroe	0.21	0.00	0.01	0.35	0.00
John Quincy Adams	0.31	0.10	0.00	0.18	0.00

Là je reconnais les topics mis sous forme de bi-grams.

for the term it is grouped together with. If a term occurs frequently in two topics, then those topics are likely related. Here is an excerpt of an H Matrix of

08/06/2022

Bonjour, aujourd'hui nous allons continuer à analyser ce bel article afin d'essayer de comprendre le maximum de choses.

Je me demande comment ils calculent les probabilités qui composent nos matrices. Peut être que sur le nombre total de mots dans un document donné ils comptent combien de fois celui-ci apparaît. Ce nombre est ensuite divisé par le nombre total de mot dans le document.

In the *H matrix*, each row represents a semantic feature that is composed of term frequencies. Each column represents a visible variable. Two terms that occur frequently together form a topic, and each term gives more contextual meaning for the term it is grouped together with. If a term occurs frequently in two topics, then those topics are likely related. Here is an excerpt of an *H Matrix* of Presidential Inauguration Speech Text:

Imaginons que les termes banane, gâteau et ananas reviennent souvent ensemble. Ça voudrait dire que banane-gâteau

forment un topic mais gâteau-ananas et banane-ananas aussi ? A partir de quel moment peut-on dire qu'un groupe de terme revient « souvent » ?

	constitution	union	peace	freedom	principle	man	spirit	party	congress	justice	year	day	policy	president	service	administration
Topics																
Union + Constitution	0.55	0.01	0.09	0.05	0.17	0.07	0.08	0.16	0.16	0.05	0.11	0.04	0.10	0.08	0.07	0.15
Man + Freedom	0.00	0.05	0.23	0.24	0.07	0.26	0.10	0.00	0.00	0.07	0.09	0.08	0.00	0.10	0.01	0.03
Business + Policy	0.05	0.00	0.15	0.05	0.04	0.02	0.04	0.15	0.15	0.13	0.08	0.04	0.16	0.02	0.08	0.08
Principles + Improvement	0.09	0.06	0.11	0.03	0.15	0.01	0.09	0.05	0.04	0.05	0.03	0.04	0.06	0.02	0.10	0.06
Family + Jobs	0.00	0.01	0.02	0.09	0.02	0.05	0.08	0.03	0.03	0.06	0.06	0.15	0.00	0.13	0.07	0.00

Excerpt of H Matrix by Anupama Garla

Autre questionnement : Dans ce tableau, comment obtient-on nos nouvelles valeurs de topic par mot ? J'ai une idée. Pour chaque topic, on regarde à travers tous les documents où il se trouve et on compte le nombre de fois où chaque mot apparaît.

Parenthèse méthodologique : est-ce que j'avance assez avec ma façon de lire l'article, de faire des déductions puis de continuer la lecture ?

Bon je suis allé voir Luan qui m'a donné quelques explications sur l'exécution des codes python qu'on nous a donné. Il m'a parlé de l'exécution et de la nécessité de créer des environnements. C'est une étape que je ne comprenais pas en entier mais que je ne pensais pas nécessaire. J'ai pu exécuter entièrement le code des « 20news ».

Retournons voir notre article.

The Art of Topic Modeling

The output of NMF changes each time you run it, and the topics are not resolved — the data scientist must infer the topic from the highest word frequencies per topic, using the H matrix. This is where the art of choosing the correct number of topics comes into play. Too many, and you have topic repeats or topics composed of sparsely related words. Too few, and you have not very meaningful topics.

NMF. Le F veut dire factorisation. On fait donc une factorisation « non-négative » de la matrice de base. C'est ce qui permet d'obtenir les deux autres matrices ?

Il faut en tout cas choisir le nombre de topic pour en avoir assez mais pas trop en fonction de nos données.

NMF Math

Like most machine learning algorithms, NMF operates by starting with a guess of values for W and H, and iteratively minimizing the loss function. Typically it is implemented by updating one matrix (either W or H) for each iteration, and continuing to minimize the error function, $\|V - WH\| = 0$, while W and H

Intéressant

On met à jour l'une des matrices seulement pendant qu'on s'assure que cela diminue la « fonction d'erreur » (error function).

continuing to minimize the error $\|V - WH\| = 0$, while W and H values remain non-negative, until W and H are stable. There are different loss functions used to update the matrices based on the specific programming package you use, but the one proposed in the original Lee and Seung paper is the *multiplicative update rule*. Honestly the math is fairly complex, and it would take me some study to wrap my head around it and explain it in a no-fuss style. I'm going to call it out-of-scope of this article. :-).

Très intéressant.

« loss functions » et « multiplicative update rule ».

Je me demande si « loss function » c'est la même chose que « error function ».

NMF caveats

NMF requires that all documents are fairly similar — for instance if you are comparing faces, you would look at faces from a similar angle. If you are comparing texts, you would look at texts of similar lengths. For more complex documents/images, you would need more layers of hidden variables, and that project would venture into the arena of neural networks.

Intéressant

NMF and TF-IDF



The advantage of NMF, as opposed to **TF-IDF** is that NMF breaks down the V matrix into *two smaller matrices*, W and H . The data scientist can set the

TF-IDF est une sorte d'autre méthode de topic modeling ?

Python Implementation

This is generally the order of operations:

1. Import Relevant Libraries to Jupyter Notebook / Install Libraries in your Environment
2. Select Data and Import
3. Isolate Data to Topic Model
4. Clean Data
5. Create Function to Pre-process Data*
6. Create Document Term Matrix ' V '
7. Create Function to Display Topics*
8. Run NMF on Document Term Matrix ' V '
9. Iterate until you find useful Topics

J'espère pour demain exécuter plusieurs codes et explorer cette histoire de directory de plus près.

09/06/2022

Bonjour tout le monde.

Où en étions-nous ? Ah oui, les programmes à exécuter.

Bon je n'arrive pas vraiment à me concentrer là-dessus. Faisons quelque chose d'utile en attendant.

Bon bah commençons par voir ce qu'est une « loss function » finalement.

In mathematical optimization and decision theory, a **loss function or cost function** (sometimes also called an **error function**)^[1] is variables onto a **real number** intuitively representing some "cost" associated with the event. An **optimization problem** seeks to minir

Donc la « loss function » et « error function » sont bel et bien les mêmes.

t. **An optimization problem seeks to minimize a loss function.** Ai

Aah il se peut que je l'aie déjà croisé quelque part cette fonction de perte puisque j'ai dû faire de l'optimisation.

e concept, as old as Laplace, was reintroduced in statistics by Abraham Wald in the ret. **In classification, it is the penalty for an incorrect classification of an example.** In ce the works of Harald Cramér in the 1920s.^[3] In optimal control, the loss is the pen:

Mhmm, intéressant. C'est donc très utilisé.

Je vais voir cette video tiens : <https://www.youtube.com/watch?v=QBbC3Cjsnjg>

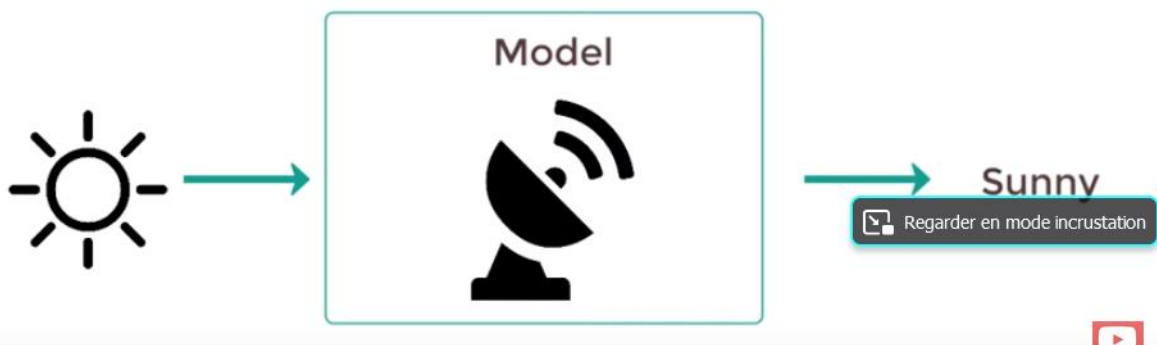
Regression

3. **Pseudo-Huber Loss** $= \begin{cases} (y - \hat{y})^2 & ; |y - \hat{y}| \leq \alpha \\ |y - \hat{y}| & ; otherwise \end{cases}$ ←

Classification

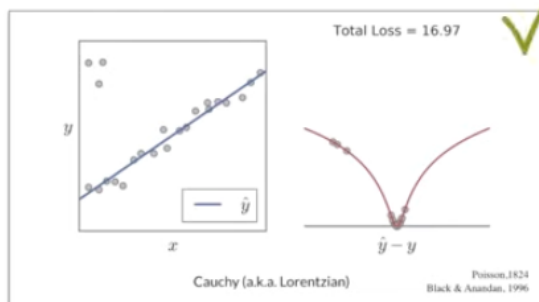
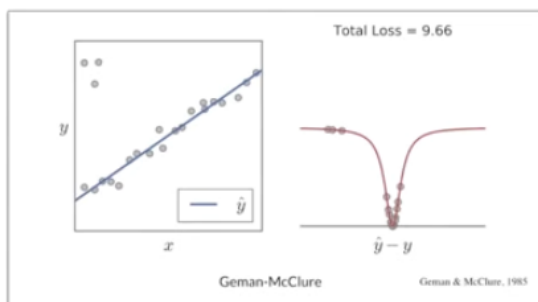
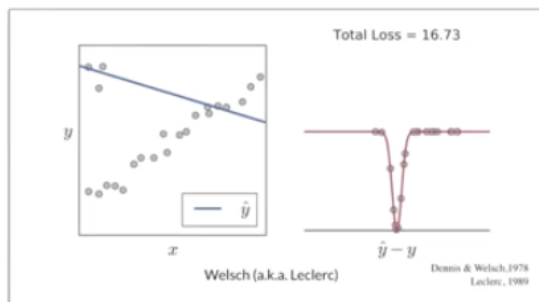
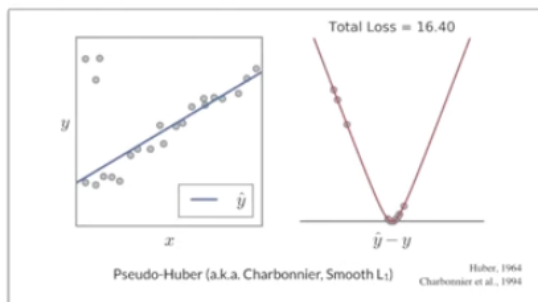
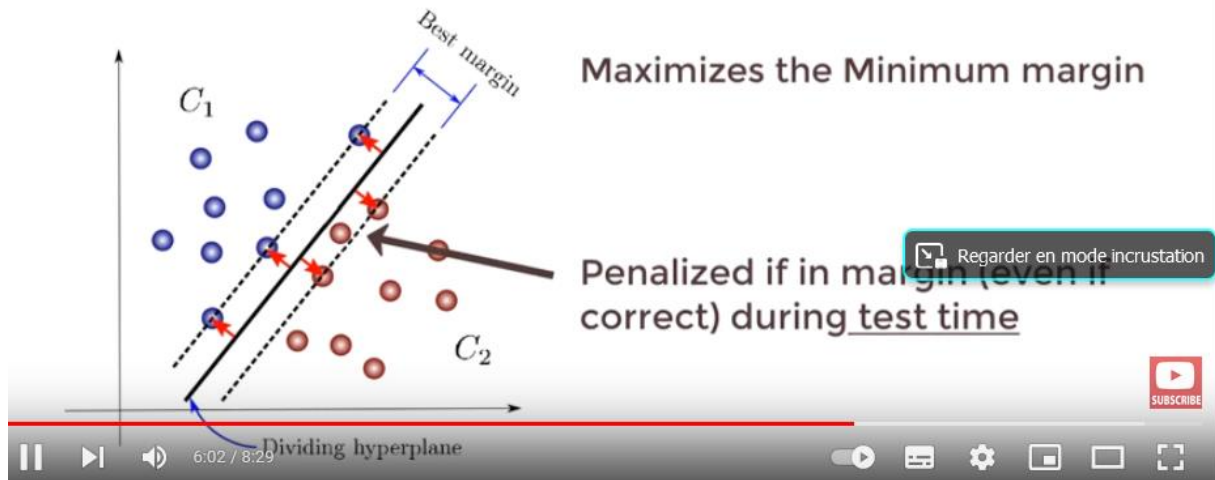
1. **Cross Entropy Loss**

2. **KL Divergence** $= \overset{\text{cross}}{\text{entropy}} H(p, q) - \overset{\text{entropy}}{H(p)}$

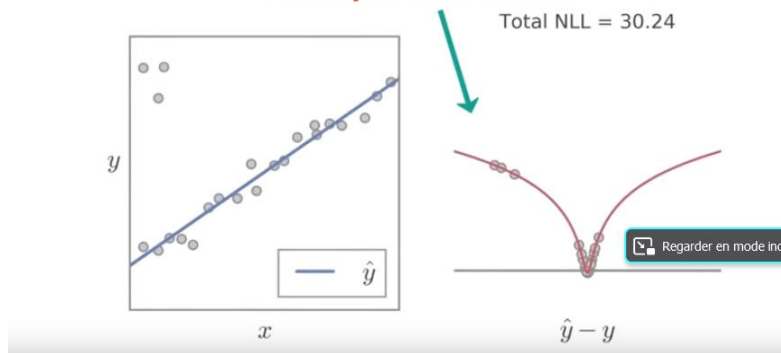


Classification

3. Hinge Loss - Used in Support Vector Machines



Adaptive Loss



C'était très intéressant, on dirait des droits de regression.

Voyons maintenant cette video : <https://www.youtube.com/watch?v=-qT8fJTP3Ks>

```
model.compile(loss='binary_crossentropy',  
              optimizer='Adam',  
              metrics=['accuracy'])
```



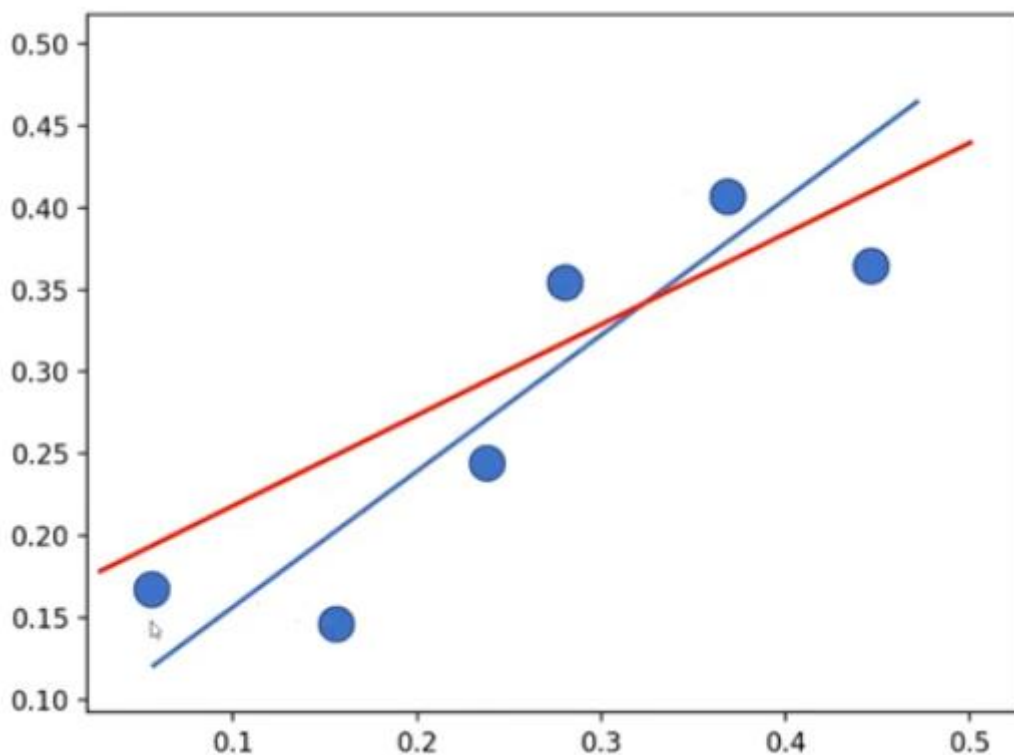
Beurk

A loss function / cost function / error function

Quantifies the error between output of the algorithm and given target value.

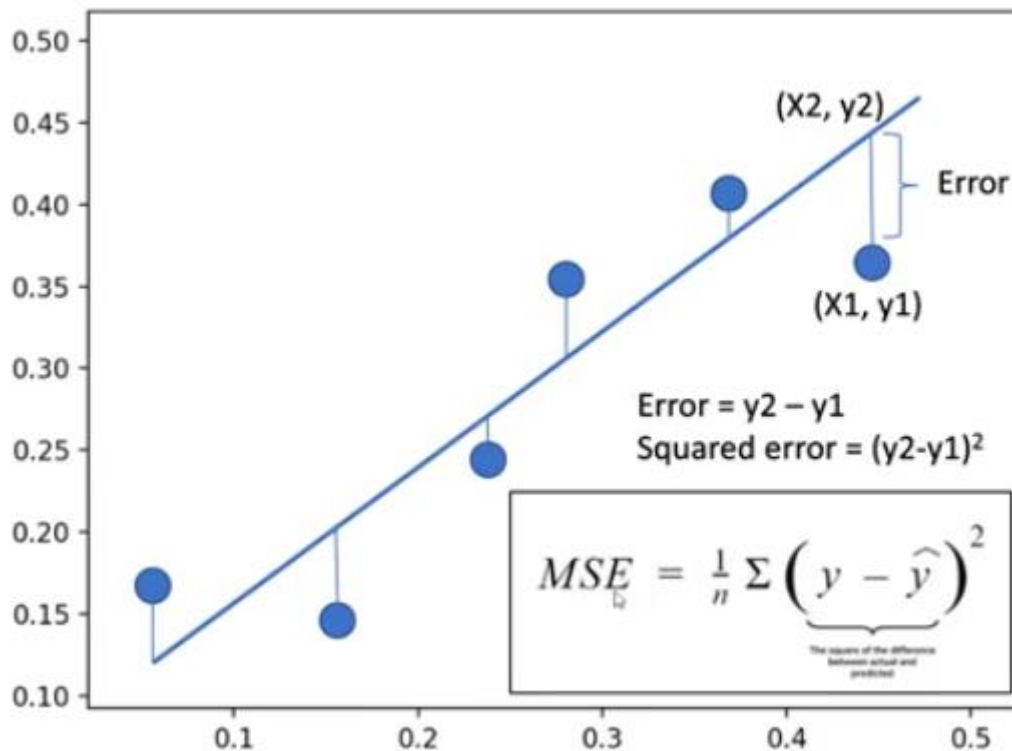
Il nous parle de training our model

Quantifies de error between the output and the expected value



Le rouge prédit mieux certaines données

Le bleu en prédit mieuc d'autres



Square it positive/negative side don't matter

Mse mean squared error

Common loss functions in machine learning

Regression

Mean squared error

$$MSE = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}$$

Others:

Mean absolute error and mean bias error

Classification

Binary and categorical cross entropy

$$CE = - \sum_i^C t_i \log(s_i)$$

In a **binary classification problem**, where $C=2$, the Cross Entropy Loss can be defined also as

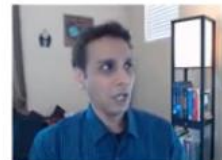
$$CE = - \sum_{i=1}^{C=2} t_i \log(s_i) = -t_1 \log(s_1) - (1 - t_1) \log(1 - s_1)$$

CCE: Train a network to output a probability over the C classes for each image. Good for multiclass problems.

BCE: sets up a binary classification problem between $C=2$ classes for every class in C . Good for multi-label problems.

Others:

Hinge loss / SVM loss.



https://gombru.github.io/2018/05/23/cross_entropy_loss/

Une autre video : <https://youtu.be/aJToUocPLg4>

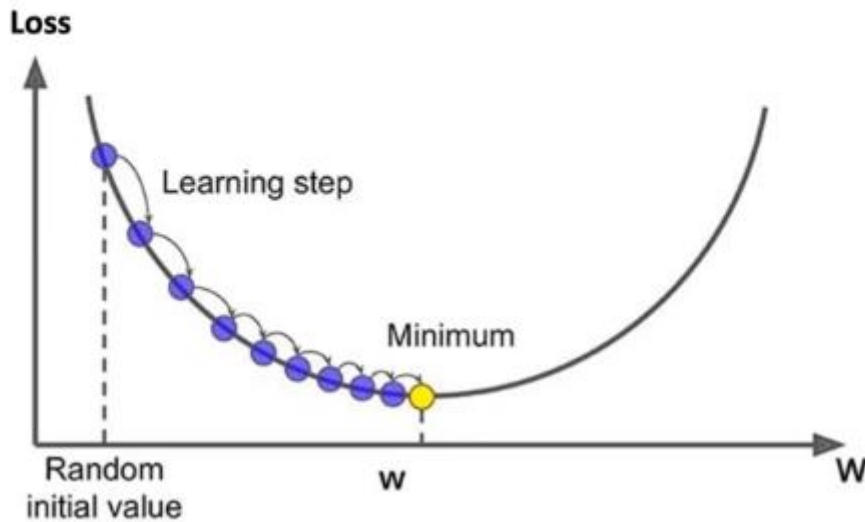
Très intéressant mais je n'ai pas retenu grand-chose.

Et une autre encore : <https://youtu.be/JhQqquVeCE0>

**Optimizers update the model in response to the output of the loss function.
Optimizers assist in minimizing the loss function.**

Comme en modèle linéaire/régression linéaire

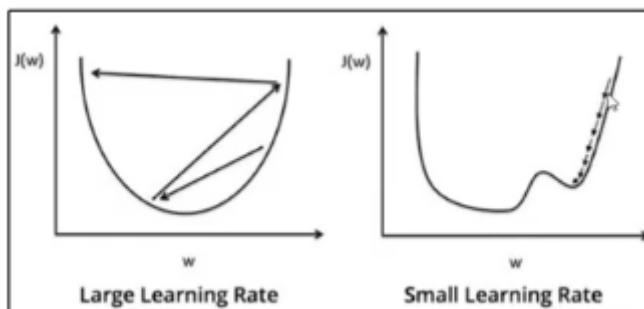
Gradient descent



Next step à gauche ou droite puis calcul de loss function pour savoir si on va dans la bonne direction pour trouver le minimum

Faire des grand ou petit pas

Learning rate = la taille des pas



Alors ça c'est drôle, tu peux te tromper de minimum

Pseudo-minimum

Adam optimizer

Original paper:

<https://arxiv.org/abs/1412.6980>

... the name Adam is derived from adaptive moment estimation.

Mhhhm interessant

Passons maintenant à l'exécution de « methods_headlines_dataset » :

In this notebook, we run NMF, NCPD, LDA, and online NCPD as dynamic topic modeling on the 'A Million News Headline' dataset downloaded from [Kaggle](#).

Cette fois ci j'ai essayé de faire :

```
conda env create -f environment.yml
```

Car j'avais cette erreur :

```
-----
ModuleNotFoundError                                Traceback (most recent call last)
<ipython-input-2-b9b069ee3627> in <module>
    17 import matplotlib.pyplot as plt
    18 from config import data_dir, results_dir
--> 19 from covid19 import plotting, utils
    20 from covid19.online_CPDL.ocpdl import Online_CPDL

ModuleNotFoundError: No module named 'covid19'
```

Il semble que ce ne soit pas fonctionnel après cette simple commande même s'il semble m'avoir installé plein de choses étranges. Des librairies je crois :

sqlite-3.38.3	806 KB	#####	100%
regex-2022.3.15	320 KB	#####	100%
prompt-toolkit-3.0.2	259 KB	#####	100%
qtconsole-5.3.0	93 KB	#####	100%
nltk-3.7	1010 KB	#####	100%
certifi-2022.5.18.1	157 KB	#####	100%
pyrsistent-0.18.0	87 KB	#####	100%
pandocfilters-1.5.0	11 KB	#####	100%

Voilà à quoi ça ressemblait.

```
s (from cycler>=0.10->matplotlib) (1.15.0)
> conda activate dynamic_topic_modeling
> |
```

Ajoutons ceci pour voir

```
ng> pre-commit install
applet de commande, fonction, fichier de script ou programme exécuté
le chemin d'accès est correct et réessayez.

[!] CommandNotFoundException
```

La chose ci-dessus ne fonctionne pas vraiment

```
Entrée [ ]: rank = 25
            n_top_words = 5 # number of keywords to display for each topic
            save_figures = True # True to save results; False otherwise
            load_NMF_factors = False # True for loading pre-saved NMF factors; False
```

Intéressant. On retrouve quelque chose qui ressemble à ce qu'on a vu sur le site de Garla.

Load ABC news dataset

```
Entrée [7]: data = pd.read_csv(directory, error_bad_lines=False, warn_bad_lines = True);  
data.head()
```

Out[7]:

	publish_date	headline_text
0	20030219	aba decides against community broadcasting lic...
1	20030219	act fire witnesses must be aware of defamation
2	20030219	a g calls for infrastructure protection summit
3	20030219	air nz staff in aust strike for pay rise
4	20030219	air nz strike to affect australian travellers

Ah je reconnais cette fonction qui aide a visualiser les premières lignes de donnée

```
Entrée [9]: # Format the date of publishing  
data['publish_date'] = pd.to_datetime(data['publish_date'], format='%Y%m%d')  
data.head()
```

Out[9]:

	publish_date	headline_text
0	2003-02-19	aba decides against community broadcasting lic...
1	2003-02-19	act fire witnesses must be aware of defamation

Là je crois qu'on converti le format de la date mais je me demande pourquoi. Est-ce juste une question esthétique (lisibilité) ou pour que le programme fonctionne ?

```
Entrée [12]: # Sort data by chronological order  
data_subsample.sort_values(by=['publish_date'], inplace=True)  
data_subsample.head()
```

Out[12]:

	publish_date	headline_text
117	2003-02-19	omodei to stay in politics
195	2003-02-19	williams says tight bowling key to warriors win

On trie chronologiquement les données

```
Entrée [14]: # Create a yyyy-mm column  
data_subsample['month_year'] = pd.to_datetime(data_subsample['publish_date']).dt.to_period('M')  
data_subsample.head()
```

Out[14]:

	publish_date	headline_text	month_year
117	2003-02-19	omodei to stay in politics	2003-02
195	2003-02-19	williams says tight bowling key to warriors win	2003-02
423	2003-02-19	williams says tight bowling key to warriors win	2003-02

Ok

```
Entrée [15]: # Test example for monthly subsampling  
len(data_subsample[data_subsample["month_year"] == pd.to_datetime('200709', format='%Y%m').to_period('M')])
```

Out[15]: 700

Extract TF-IDF weights and features

```
Entrée [*]: # Vary the choices of the parameters of TfidfVectorizer
n_features = 7000
max_df=0.7
min_df=5
```

Je me demande ce qu'il extrait vraiment. C'est quoi ces TF-IDF poids et « features ».

Extract TF-IDF weights and features

```
Entrée [17]: # Vary the choices of the parameters of TfidfVectorizer
n_features = 7000
max_df=0.7
min_df=5
stop_words_list = nltk.corpus.stopwords.words('english')
stop_words_list.append("abc") # remove the common string "abc"
```

Ça c'est la partie paramétrage

```
vectorizer = TfidfVectorizer(max_df=max_df,
                             min_df=min_df,
                             #token_pattern = '[a-zA-Z]+',
                             max_features=n_features,
                             #ngram_range = (1,2),
                             stop_words=stop_words_list)
```

Ils créent ensuite un objet vectorizer de type TfidfVectorizer qui va prendre en entrée tout ce qu'on a créé comme paramètre juste avant.

D'ailleurs pourquoi le df minimum est plus grand que le maximum ? **Mystère**

```
vectors = vectorizer.fit_transform(corpus)
feature_names = vectorizer.get_feature_names()
dense = vectors.todense()
denselist = np.array(dense).transpose()
```

On fait subir des transformations à cet objet jusqu'à ce qu'on obtienne une grosse matrice. La fameuse matrice contenant les mots par document j'imagine :

```
Entrée [18]: print("Shape of the matrix {}".format(denselist.shape))
print("Dimensions: (vocabulary, headlines)")
```

```
Shape of the matrix (7000, 150500).
Dimensions: (vocabulary, headlines)
```

10/06/2022

Bonjour !

Commençons par le code aujourd'hui. Nous allons voir jusqu'où on va pouvoir l'exécuter. J'espère ensuite pouvoir continuer avec le « tuto » de madame Anupama Garla et pourquoi pas faire notre propre programme de topic modeling.

```
17]: # Organize data into a third-order tensor
num_time_slices = dfg.ngroups # number of time slices
n_div = 1 # number of months per time slice
n_headlines = sample_val*n_div # number of headlines per time slice
data_tens = fold(denselist, 1, (denselist.shape[1] // n_headlines, denselist.shape[0], n_headlines))

print("Shape of the tensor {}".format(data_tens.shape))
print("Dimensions: (time, vocabulary, headlines)")
```

```
Shape of the tensor (215, 7000, 700).
Dimensions: (time, vocabulary, headlines)
```

Mhmm, donc là on fait un tensor ? D'accord.

Bon, il faut savoir ce que fait la fonction fold() afin de comprendre comment se construit se tensor.

Il a besoin de cette chose nommée « denselist » qui pour moi est notre matrice a 2 dimension contenant les mots par documents. En suivant cette logique on peut se dire que pour en faire un tensor de dim 3 il faut ajouter la dimension temporelle.

```
print("Dimensions: (time, vocabulary, headlines)")
type(denselist)
```

```
Shape of the tensor (215, 7000, 700).
Dimensions: (time, vocabulary, headlines)
```

```
]: numpy.ndarray
```

Là j'ai voulu regarder de quel type était denselist. J'aurai aimé faire denselist.head() mais on ne peut pas faire ça avec, ça ne fonctionne pas. Du coup c'est un tableau numpy. Je pense.

Bon ne trainons pas et continuons l'exécution.

```
[23]: # List of all features/words extracted
feature_names
```

```
'argentina',
'arm',
'armed',
'armidale',
'arms',
'armstrong',
'army',
```

Avec tout ça j'ai oublié ce qu'était un « feature »

```
features/words extracted
```

Ah d'accord, c'est des mots qu'il extrait. Mais pourquoi faire à cette étape ci ?

Après il faut se rappeler qu'on est dans une partie du programme nommée « extract TF-IDF weights and features ».

Remarque : Je n'ai beaucoup de recul en ce moment. Peut-être que ça viendra. Je verrai peu à peu comment chacune de ces petites actions s'inscrit dans l'ensemble.

Plotting Code

```
[ ]: sns.set(style="whitegrid", font_scale=1.001, context="talk")

def heatmap(
    data,
    x_tick_labels=None,
    x_label="",
    y_tick_labels=None,
    y_label="",
    figsize=(7, 9),
    max_data=None,
```

Oh on prépare des graphiques.

On a créé une fonction qui nous aide à générer nos « heatmaps ». On utilise dedans plein de fonctions que je ne connais pas vraiment, je peux essayer de deviner à quoi ils servent en lisant leurs noms.

Passons.

Run NMF

```
[ ]: # Run NMF
if load_NMF_factors:
    W, H = pickle.load(open(os.path.join(local_path, "NMF_factors_headlines.pickle"), "rb"))
else:
```

Nous voici à la partie Run NMF. On va donc y faire la NMF. Rappelons qu'on travaille sur des « news headlines » que je traduis par « titres d'article de presse ».

Cette partie du programme se décompose en 5 blocs qu'ils ont pris le temps de nommer respectivement :

- # Run NMF
- # Display topics
- # Topic Keywords representations (intéressant)
- # Get topic distributions for each time slice
- #Visualize topic distributions

Ils contiennent aussi des sous-parties

Run NMF

```
[*]: # Run NMF
if load_NMF_factors:
    W, H = pickle.load(open(os.path.join(local_path, "NMF_factors_headlines.pickle"), "rb"))
else:
    nmf = NMF(n_components=rank, init='nndsvd')
    W = nmf.fit_transform(denselist) # Dictionary
    H = nmf.components_ # Topic representations
    NMF_factors = W,H
    with open(os.path.join(local_path, "NMF_factors_headlines.pickle"), "wb") as f:
        pickle.dump(NMF_factors, f)
```

Il prend tout son temps pour exécuter cette partie. On m'avait dit pourtant que ce programme était plus long à exécuter.

Je propose que l'on fasse autre chose en attendant que mon ordi surpuissant fasse le travail.

Regardons le code d'Anupama Garla (j'adore ce nom) ou bien des vidéos utiles.

Finalement je suis allé voir avec Luan comment les programmes fonctionnent (pas) sur son ordi.

Bon Luan a finalement réussi à faire fonctionner les programmes. Il a dû installer « latex » et le mettre à jour je crois mais je ne suis pas sûre.

Je viens de créer un fichier NMF_tentative_1.py afin d'essayer d'en faire une à l'aide du tutoriel de Garla.

Je remets le lien ici : <https://towardsdatascience.com/nmf-a-visual-explainer-and-python-implementation-7ecdd73491f8>

A mesure que j'avance dans le tutoriel de Garla je me rends compte qu'il y a une étape que l'on n'a pas eu dans les programmes précédents. Il s'agit du « text cleaning ». J'ai vu un autre stagiaire qui lui travaillait sur la LDA en faire.

Bon il faut y aller. C'est fini pour aujourd'hui.

13/06/2022

Continuons ce que nous faisons avant ce week-end.

Cela fait un moment que la NMF que j'ai relancé continue de tourner

Presidential Inaugural Addresses

US president name, date, and speech text



Data Code (12) Discussion (1) Metadata

About Dataset

Context

Every Presidency starts off with the Inaugural Address. This defines the course for the next 4 years. How do the length, word usage, lexical diversity change from President to President?

Les données du tutoriel à télécharger

14/06/2022

```
-----  
MemoryError                                Traceback (most recent call last)  
<ipython-input-19-2d4209c0f51e> in <module>  
      4 else:  
      5     nmf = NMF(n_components=rank, init='nndsvd')  
----> 6     W = nmf.fit_transform(denselist) # Dictionary  
      7     H = nmf.components_ # Topic representations  
      8     NMF_factors = W,H  
  
C:\Anaconda3\lib\site-packages\sklearn\decomposition\_nmf.py in fit_transform(self, X, y, W, H)  
    1317         shuffle=self.shuffle)  
    1318  
-> 1319         self.reconstruction_err_ = _beta_divergence(X, W, H, self.beta_loss,  
    1320                                                    square_root=True)  
    1321  
  
C:\Anaconda3\lib\site-packages\sklearn\decomposition\_nmf.py in _beta_divergence(X, W, H, beta, square_root)  
    106         res = (norm_X + norm_WH - 2. * cross_prod) / 2.  
    107     else:  
-> 108         res = squared_norm(X - np.dot(W, H)) / 2.  
    109  
    110         if square_root:  
  
MemoryError: Unable to allocate 7.85 GiB for an array with shape (7000, 150500) and data type float64
```

Après quelques heures de fonctionnement il me dit qu'il n'a pas pu allouer la place nécessaire au tableau de données

16/06/2022

5. Create Pre-process Data*

Here we will **lemmatize** all the words in the speeches so that different forms of a particular word will be reduced to a base form, for instance noun plurals become singular and all verb tenses become present. This simplifies the text so that repeated instances of slight variations of a word are interpreted as one word. One can stem or lemmatize, more info [here](#). The other thing we are doing here is

environ 77 100 résultats (0,42 secondes)

La lemmatisation désigne un traitement lexical apporté à un texte en vue de son classement dans un index ou de son analyse.

<https://fr.wiktionary.org/wiki/tokenize> ▼

tokenize - Wiktionnaire

tokenize. (Informatique) Parser un texte composé de tokens. Tokenizing is the process of converting a string into a list of substrings, known as tokens.

Il faut revoir ce qu'est la SVD afin de mieux comprendre la NMF

On m'a dit que lorsqu'on fait une NMF on fixe alternativement l'une des matrices tandis qu'on transforme l'autre par rapport au premier. Puis on inverse les rôles. Je ne sais cependant pas quel genre de transformation est appliquée.

17/06/2022

Bonjour,

Aujourd'hui il n'y avait pas de connexion sur les ordinateurs de l'université. J'ai dû retourner jusque chez moi. Cela m'a par contre permis de prendre le temps d'installer plusieurs logiciels sur mon ordinateur fixe.

Je vais préparer avec Luan un diapo à envoyer pour la présentation lors de la journée des stagiaires.

<https://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-explo-nmf.pdf>

2.1 Principes

Soit \mathbf{X} une matrice $(n \times p)$ ne contenant que des valeurs non négatives et sans ligne ou colonne ne comportant que des 0 ; r un entier choisi relativement petit devant n et p .

La factorisation non-négative de la matrice \mathbf{X} est la recherche de deux matrices $\mathbf{W}_{n \times r}$ et $\mathbf{H}_{r \times p}$ ne contenant que des valeurs positives ou nulles et dont le produit approche \mathbf{X} .

$$\mathbf{X} \approx \mathbf{WH}.$$

Okay

Le choix du *rang* de factorisation $r \ll \min(n, p)$ assure une réduction drastique de dimension et donc des représentations parcimonieuses. Évidemment, la qualité d'approximation dépend de la parcimonie de la matrice initiale.

La factorisation est résolue par la recherche d'un optimum local du problème d'optimisation :

$$\min_{\mathbf{W}, \mathbf{H} \geq 0} [L(\mathbf{X}, \mathbf{WH}) + P(\mathbf{W}, \mathbf{H})].$$

On approxime quoi exactement ? la matrice de départ ?

$$\min_{\mathbf{W}, \mathbf{H} \geq 0} [L(\mathbf{X}, \mathbf{WH}) + P(\mathbf{W}, \mathbf{H})].$$

L est une fonction perte mesurant la qualité d'approximation et P une fonction de pénalisation optionnelle ; L est généralement soit un critère de moindres carrés (LS ou norme de Frobenius des matrices ou "norme trace"), soit la divergence de Kullback-Leibler (KL) ; P est une pénalisation optionnelle de régularisation utilisée pour forcer les propriétés recherchées des matrices W et H , par exemple, la parcimonie des matrices ou la régularité des solutions dans le cas de données spectrales.

$$LS : L(A, B) = \text{tr}((A - B)(A - B)') = \sum_{i,j} (a_{i,j} - b_{i,j})^2,$$

$$KL : L(A, B) = KL(A||B) = \sum_{i,j} a_{i,j} \log\left(\frac{a_{i,j}}{b_{i,j}}\right) - a_{i,j} + b_{i,j}.$$

Bon nous reviendrons sur ceci plus tardivement car c'est intéressant

Je vais vous montrer le PDF que nous avons préparé pour notre court passage oral

Dechery Luan
Ndiaye Abdourahmane

Topic Modeling : Analyse de tweets



Méthodes:

- NMF : Non-negative matrix factorization
- NCPD : Nonnegative CANDECOMP/PARAFAC tensor decomposition
- ONCPD : Online Nonnegative CANDECOMP/PARAFAC tensor decomposition

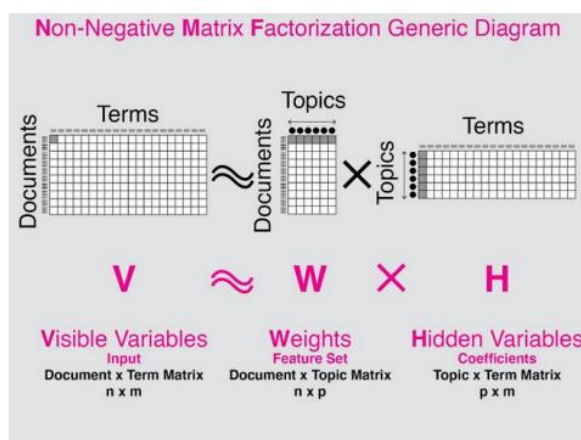
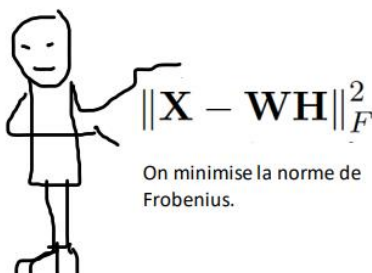


université
lumière
LYON 2

La NMF

Dans un corps de texte:

- Les même mots vont apparaitre à la même fréquence pour décrire un sujet.
- On peut donc détecter le sujet en fonction de la fréquence d'apparition de certains mots.



Matrix Decomposition in NMF Diagram by Anupama Garla

<https://towardsdatascience.com/nmf-a-visual-explain-python-implementation-7ecdd73491f8>

université
lumière
LYON 2

Exemple
graphique de
résultat



Merci de votre attention.

université
lumière
Lyon 2

21/06/2022

Bonjour, hier c'était la journée des stagiaires

Aujourd'hui continuons notre tutoriel et structurons notre rapport de stage

Please use the NLTK Downloader to obtain the resource:

```
>>> import nltk
>>> nltk.download('punkt')
```

For more information see: <https://www.nltk.org/data.html>

Attempted to load **tokenizers/punkt/english.pickle**

```
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
```

6. Create Document Term Matrix 'V'

Here I added some stopwords to the stopwords list so we don't get words like 'America' in the topics as that is not super meaningful in this context. I also use TF-IDF Vectorizer rather than a simple Count Vectorizer in order to give greater value to more unique terms. You can learn more about TF-IDF [here](#).

<https://towardsdatascience.com/tf-idf-a-visual-explainer-and-python-implementation-on-presidential-inauguration-speeches-2a7671168550>

(gestion de notes et rattrapages)

7. Create Function to Display Topics*

To evaluate how useful the topics created by NMF are, we need to know what they are. Here I create a function to display the top words activated for each topic.

Il y a vraiment plein de façon de coder dans python que je ne connais pas

22/06/2022

Il est temps de se pencher un peu plus sur le rapport de stage

Cherchons comment il se structure sur internet

23/06/2022

Je n'avais pas accès à l'ordinateur dans la salle où tout ce qu'il faut est installé

24/06/2022

Aujourd'hui Mr Chrétien m'a expliqué ce qu'était latex et m'a recommandé de faire mon rapport de stage avec. Je serai ravi d'essayer.

[https://fr.overleaf.com/learn/latex/Learn LaTeX in 30 minutes](https://fr.overleaf.com/learn/latex/Learn_LaTeX_in_30_minutes)

- Nous avons trouvé un tutoriel rapide afin de voir comment faire du LATEX.

J'aimerais commencer par faire de la programmation et finir d'analyser les données de discours présidentiels.

L'après-midi en changeant de salle je pourrai explorer le LATEX.

Allons faire le tutoriel.

Champs

Type de colonne :


	Standard	Standard	Standard	Standard	Standard
1		Name	Inaugural Address	Date	text
2	4	George Washington	First Inaugural Address	Thursday, April 30, 1789	Fellow-Citizen
3	5	George Washington	Second Inaugural Address	Monday, March 4, 1793	Fellow Citizen
4	6	John Adams	Inaugural Address	Saturday, March 4, 1797	WHEN it was
5	7	Thomas Jefferson	First Inaugural Address	Wednesday, March 4, 1801	Friends and I
6	8	Thomas Jefferson	Second Inaugural Address	Monday, March 4, 1805	PROCEEDING,
7	9	James Madison	First Inaugural Address	Saturday, March 4, 1809	UNWILLING I
8	10	James Madison	Second Inaugural Address	Thursday, March 4, 1813	ABOUT to ac
9					

Voici à quoi ressemblaient les données avant traitement. Moi qui pensais qu'il n'y avait que deux colonnes. Il va donc falloir se débarrasser des colonnes en trop. Mais aussi ne garder que les discours de premier mandat. Par contre je ne suis pas sûre de ce qu'on fera des dates.


```
df = df[['Name', 'text']]
# J'imagine qu'on ne conserve que ces deux colonnes de donnée
```

On dirait qu'on s'en passe pour l'instant

4. Clean Data

I want to make all of the text in the speeches as comparable as possible so I will create a cleaning function that removes punctuation, capitalization, numbers, and strange characters. I use regular expressions  which offers a lot of ways to 'substitute' text. There are tons of [regular expression cheat sheets](#), like this [one](#). I 'apply' this function to my speech column.

Regular expressions (regex or regexp) are extremely useful in [extracting information from any text](#) by searching for one or more matches of a specific search pattern (i.e. a specific sequence of ASCII or unicode characters).

 One of the most interesting features is that [once you've learned the syntax, you can actually use this tool in \(almost\) all programming languages](#) (JavaScript, Java, VB, C#, C / C++, Python, Perl, Ruby, Delphi, R, Tcl, [and](#) many others) with the slightest distinctions about the support of the most advanced features and syntax versions supported by the engines.

Donc j'ai déjà vu des expressions régulières lors du projet en Bash