



# EXTRACTION DE THÈMES ET CLASSIFICATION

PSC Rapport Final

Avril 2023

---

Majd Abid, Eyal Benaroche, Burhan Hammadia, Nazim Messied, Victor Zhuang



## REMERCIEMENTS

---

Nous tenons à tout d'abord remercier Monsieur Émile Lucas, notre tuteur technique qui nous a aidé et supervisé tout au long du projet.

Nous remercions également chaleureusement Madame Amélie Gagnon pour son aide précieuse sur les données traitées.

Enfin, nous voudrions remercier Monsieur El Mahdi El Mhamdi pour ses conseils avisés.

## CONTRIBUTIONS

Dans ce projet, nous avions comme objectif d'apporter une contribution conséquente dans l'analyse de large corpus de textes venant de Plan de l'Éducation Nationale de plusieurs pays dans le monde sur plusieurs périodes. Pour atteindre cet objectif, nous avons utilisé une large variété de techniques comme *GloVe*, *TF-IDF*, *Topic Modeling* ou encore des *Réseaux de Neurones Récurrents (RNN)*. Notre approche impliquait une analyse approfondie et complète du corpus pour pouvoir en extraire les informations les plus pertinentes.

L'une des contributions principales de notre travail est l'extraction et l'analyse des thèmes présents dans le corpus. En utilisant les techniques mentionnées précédemment, nous avons été en mesure d'identifier les thèmes les plus pertinents qui ont émergé du corpus. Cela nous a permis d'avoir une compréhension approfondie des thèmes, notamment d'un point de vue continental, qui pourrait avoir d'importantes implications dans le planning des politiques scolaires et éducatives.

Par ailleurs, cette approche multiforme nous a permis d'obtenir des résultats assez robustes et interprétables, ce qui a contribué au domaine du *Natural Language Processing (NLP)* et *Machine Learning* en mettant en valeurs l'efficacité des techniques utilisées dans l'analyse de large corpus de textes. Nos résultats montrent également le potentiel de ces techniques dans l'extraction d'informations pertinentes dans des données textuelles complexes, qui pourrait être appliqué dans différents domaines, comme ici l'éducation, ou alors la linguistique et les sciences sociales.

En outre, nous avons également développé une méthode basée sur la *Named Entity Recognition (NER)* pour améliorer la qualité du traitement des textes et l'analyse du corpus. Cette technique a permis une meilleure identification et classification des entités présentes dans le texte, mais également un meilleur temps d'exécution de l'algorithme, ce qui a permis une analyse plus fine et précise du corpus. En somme, notre contribution repose sur une approche hybride qui combine différentes techniques de traitement de données textuelles pour extraire des informations pertinentes et utiles. Ces résultats pourront être utilisés par les décideurs politiques, les chercheurs et les enseignants pour une meilleure compréhension des enjeux éducatifs à l'échelle nationale et internationale.

## TABLE DES MATIÈRES

---

<b>1</b>	<b>Introduction</b>	<b>6</b>
<b>2</b>	<b>Préparation et pre-processing des données</b>	<b>7</b>
2.1	Introduction . . . . .	7
2.2	Processus . . . . .	7
2.2.1	Importation . . . . .	8
2.2.2	Résultats . . . . .	8
2.3	Analyse . . . . .	9
<b>3</b>	<b>TF-IDF</b>	<b>10</b>
3.1	Bag Of Words . . . . .	10
3.2	TF-IDF . . . . .	11
3.2.1	Term Frequency . . . . .	11
3.2.2	Term Frequency, Inverse Document Frequency . . . . .	11
3.3	Analyse . . . . .	12
3.3.1	Résultats Notables . . . . .	12
3.3.2	Dataset et Feedback . . . . .	13
<b>4</b>	<b>GloVe</b>	<b>14</b>
4.1	Introduction . . . . .	14
4.2	Fonctionnement de GloVe . . . . .	14
4.2.1	Notations . . . . .	14
4.2.2	Fonctionnement . . . . .	15
4.2.3	Minimisation de la fonction de perte : méthode de descente de gradient . . . . .	16
4.3	Applications . . . . .	16
4.3.1	Quelques exemples simples . . . . .	16
4.3.2	Application au corpus de textes . . . . .	18
<b>5</b>	<b>clustering</b>	<b>19</b>
5.1	Introduction . . . . .	19
5.2	Algorithmes . . . . .	19
5.2.1	K-Means . . . . .	19
5.2.2	DBscan . . . . .	19
5.3	Résultats & Analyse . . . . .	21
<b>6</b>	<b>Topic Modelling et REGEX</b>	<b>23</b>
6.1	Introduction . . . . .	23
6.2	Allocation de Dirichlet latente (LDA) . . . . .	23
6.2.1	Notations et définitions . . . . .	23
6.2.2	processus génératif . . . . .	24
6.2.3	Retrouver les variables latentes . . . . .	26
6.2.4	Inférence variationnelle (VI) . . . . .	27
6.3	Application à notre jeu de données . . . . .	27
6.3.1	Premiers tests . . . . .	28
6.3.2	REGEX . . . . .	29
6.3.3	Application à plus grande échelle . . . . .	30
<b>7</b>	<b>Named Entity Recognition (NER)</b>	<b>32</b>

<b>8 Classification avec réseaux de neurones récurrents (RNN)</b>	<b>34</b>
8.1 Introduction . . . . .	34
8.2 Réseaux de neurones . . . . .	34
8.2.1 Introduction . . . . .	34
8.2.2 Rétro-propagation . . . . .	35
8.3 Réseaux de neurones récurrents . . . . .	36
8.3.1 Présentation . . . . .	36
8.3.2 Problème de la disparition du gradient . . . . .	36
8.3.3 LSTMs . . . . .	37
8.4 Application au corpus . . . . .	38
8.4.1 Création du dataset . . . . .	38
8.4.2 Résultats du modèle . . . . .	38
<b>9 Conclusion</b>	<b>41</b>
<b>10 Annexes</b>	<b>43</b>

# 1

## INTRODUCTION

---

Les analystes régionaux et chercheurs de l'UNESCO-IIPE étudient particulièrement les rapports concernant les plans d'éducation nationaux et internationaux. Pour commencer l'étude d'une nouvelle région, ils commencent leur travail par lire tous les textes des années précédentes pour pouvoir en tirer des conclusions pertinentes. La valeur ajoutée de l'outil informatique serait de leur faciliter la tâche en donnant des données prémaçhées. Ainsi, à travers Illuin technology, l'UNESCO-IIPE nous a contactés pour un sujet de classification de textes.

L'idée est de déterminer les différents thèmes traités dans certains plans nationaux pour pouvoir donner une vision globale aux analystes. Un deuxième objectif aurait été d'analyser ces données en fonction du temps : en effet, nous aurions pu voir apparaître différents thèmes dans certaines régions, puis observer leur expansion dans le monde entier. Néanmoins, à cause du manque de données par période temporelle, nous n'avons pas pu observer des résultats pertinents.

On voit déjà apparaître ici l'une des principales problématiques de notre sujet : les données. Le *dataset* que nous avions à notre disposition se constitue de l'ensemble des textes de rapports sur <https://planipolis.iiep.unesco.org/>. Chaque texte est composé de plusieurs dizaines de pages, comprenant images, tableaux et autres mises en page visuelles. Même si la base *planipolis* est assez vaste, pour un ordinateur celle-ci reste assez petite. En effet, l'ensemble des données n'était clairement pas suffisant pour entraîner un modèle d'apprentissage supervisé, et nous n'avions de toute façon pas de données labellisées pour ce potentiel entraînement. Ainsi, nous nous sommes orientés vers des modèles ne nécessitant pas d'apprentissage supervisé. Enfin, nous devions trouver une méthode qui est capable de traiter des grosses quantités de données en un temps raisonnable (les rapports font parfois une centaine de pages).

Ce PSC s'articule comme un défrichage de techniques de NLP qui pourraient être utilisées pour parvenir à notre objectif final, en prenant en considération les particularités de notre set de données. Nous essaierons tout d'abord des approches naïves (Doc2Vec, TF-IDF...) pour arriver vers des modèles mathématiques plus complexes (tel que les *RNN*).

## 2

## PRÉPARATION ET PRE-PROCESSING DES DONNÉES

### 2.1 INTRODUCTION

Avant de se lancer dans le code, nous nous sommes renseignés sur les us et coutumes du NLP. Un des premiers prérequis pour l'analyse du langage est de bien comprendre comment nous communiquons nos idées naturellement.

Pour donner du sens aux mots que nous employons, nous leur fournissons un contexte : on commence une phrase avec un sujet, on place un verbe, des compléments d'objets et circonstanciels... Cependant, lorsqu'on apprend une nouvelle langue, on se rend compte que le langage n'est pas simplement une juxtaposition d'éléments au cœur du sens de notre phrase, mais il est également composé de beaucoup de mots de liaisons. Chaque nom commun est introduit par un article, des conjonctions de coordinations permettent de relier les phrases entre elles, mais ce type de mots ne donne pas réellement un sens aux phrases. Et pourtant ils constituent une part majoritaire de notre langage ! Au même titre qu'on simplifie notre langage en présence d'individus débutant dans notre langue, il s'agit dans un premier temps de simplifier les textes complexes en assortiments de mots plus simples.

Plus simple encore, puisque nous avons décidé de traiter du langage pour en extraire du sens, il s'agit donc uniquement des mots. L'ensemble des processus de mise en forme seront donc filtrés en amont.

On définit donc ici quelques notions utiles pour le prétraitement :

**Définition 2.1.0.1** (Stopwords). *Ensemble des mots du langage qui n'apportent aucune information au texte.*

*Il n'existe pas de liste universelle pour ces mots ou des règles précise pour les identifier, mais de nombreuses listes sont disponibles (et il est possible d'en créer soit même).*

**Exemple 2.1.0.2** (Stopwords). *Stopwords usuels en anglais : and, there, is, a...*

**Définition 2.1.0.3** (Stemming). *Processus de "racinisation" consistant à retirer les suffixes d'un mot pour en obtenir la racine.*

**Exemple 2.1.0.4** (Stemming). *Exemple de stemming sur le sujet du consulting :*

- consult
- consultant
- consultants → consult
- consultantative
- consulting

### 2.2 PROCESSUS

L'ensemble des méthodes proposées ont été implémentées sous *Python* à l'aide de différentes librairies (détailées par la suite) et de nos connaissances.

### 2.2.1 • IMPORTATION

Lorsqu'on traite d'un document textuel, il faut accorder une importance particulière au format. Les formats suivants sont les plus utilisés : *.doc*, *.odt*, *.docx* pour les fichiers type *Word*; les *.tex* pour les fichiers *LAT<sub>E</sub>X*, et de nombreuses autres extensions, mais la plus utilisée, tout particulièrement lorsqu'il s'agit de partager de l'information reste le *.pdf*.

Là où la lecture d'un pdf est aisée pour l'œil humain, c'est une tâche bien plus ardue pour un ordinateur. Malheureusement, l'intégralité des textes fournis sont au format *.pdf* et la première tâche est donc de les transformer dans un format plus facilement manipulable pour nos algorithmes.

Cette première étape pourrait constituer à elle seule un PSC dans le domaine de VrDU, *Visual Rich Document Understanding*, au vu des différentes méthodes employées et de la complexité de la littérature à ce sujet. Cependant, ce n'était pas le sujet de notre PSC, bien que nous devions incorporer cette première étape au processus complet, il n'était pas souhaitable de passer l'intégralité de notre temps à raffiner cette simple tâche.

Le processus est le suivant :

1. Ouverture du pdf dans python à l'aide de la librairie *pdfplumber*.
2. Filtrage des stopwords à l'aide de la librairie *nltk*.
3. Vérification de l'appartenance au dictionnaire des mots restants.
4. Stemming des mots restants toujours via *nltk*.
5. Sauvegarde des mots restants dans un fichier *.txt* pour éviter de recommencer le processus à chaque manipulation.

### 2.2.2 • RÉSULTATS

Une fois toute la *pipeline* exécutée, il est légitime de se demander quelle est l'efficacité de cette dernière. Sans toute fois lire l'ensemble des fichiers modifiés (en particulier avec des dizaines de milliers de pages...) Voilà quelques statistiques et un aperçu de l'efficacité des modifications où on aperçoit qu'en réalité, nous supprimons une grande partie de nos différents textes :

Name	Pages	Size	Removed	% Removed
netherlands	8	1977	1950	49.66
denmark_be	81	11389	9805	46.26
ireland_de	70	9926	8393	45.82
czech_repu	69	20363	16381	44.58
estonia_hi	12	3475	2738	44.07
netherlands	92	23937	18622	43.76
lithuania_	8	1686	1308	43.69
ireland_st	28	3839	2959	43.53
norway_sec	18	2892	2228	43.52
moldova_r_	42	8266	6255	43.08
hungary_pu	49	11973	8906	42.66
montenegro	39	8220	6024	42.29
strategic_	124	25105	18102	41.9
ireland_na	63	23011	16002	41.02

FIGURE 1 – Une partie du tableau statistique des importations des fichiers traitant de l'Europe

Ici la *size* correspond à *size after removal*.

## 2.3 ANALYSE

Une première tentative de modélisation, avec les processus décrits par la suite du rapport, sans traitement préalable, avait souligné l'importance du *pre-processing* : les résultats étaient inexploitables.

1. Les stopwords représentent environ 40% des documents (nous n'avons mis ici que les documents qui ont été le plus "touchés" par le pre-processing).
2. Les erreurs d'importations lorsqu'elles arrivent se répercutent sur le document le rendant parfois inexploitable dans son intégralité. Par exemple, un des éléments de formatage du texte s'est retrouvé mal importé plus de 70 fois et faussait considérablement la représentation du texte.
3. De nombreux mots sont tout de même encore mal importés, l'erreur la plus récurrente étant qu'il manque un espace entre deux mots, causant ainsi une perte d'information sensible.
4. La mise en page est très mal étudiée. Là où elle aide fortement le lecteur à comprendre la structure du texte, elle est un obstacle lors de la compréhension pour la machine avec nos algorithmes et ne convient pas du tout à l'étude. Il aurait été préférable d'avoir un jeu de données prévu pour l'occasion.

Ainsi, afin de faciliter l'exploitation de certaines des méthodes envisagées et dans un but de clarté dans la manipulation, de nombreux tests dans les parties à suivre ont été réalisées avec des *datasets* conçus pour l'occasion, résolvant donc le problème de *garbage in, garbage out*.

## 3

# TF-IDF

---

### 3.1 BAG OF WORDS

---

Une fois les documents lisible, il s'agit maintenant de représenter chaque document par un format lisible pour l'ordinateur. La méthode la plus simple et intuitive pour cela est de représenter chaque document par l'ensemble des mots qui le compose. On réalise donc un *mapping* des mots d'un document et on compte les occurrences de chaque mot du document, donnant un vecteur de mots par document.

On utilise donc la méthode de représentation simple proposée ci-dessous : le *Bag Of Words* (BOW).

**Définition 3.1.0.1** (Document). Soit  $doc$  un document :

- $\exists d \in \mathbb{N}$ , le nombre de mots uniques du document
- $\forall k \in \llbracket 1, d \rrbracket, d[w_k] =$  fréquence de  $w_k$  dans  $doc$
- $\forall k \in \llbracket 1, d \rrbracket, d[w_k] \geq 0$

Ainsi, on construit itérativement le vecteur qui représente chaque document. La dimension du vecteur (notée  $d$ ) correspond à la taille du vocabulaire utilisé pour chaque document. On note  $n$  le nombre de documents et on construit la matrice de tous les *Bag Of Words*.

**Définition 3.1.0.2** (BOW Matrix). Soit  $n$  le nombre de documents :

Soit  $d$  la taille du vocabulaire du corpus défini par

$$d = \left| \bigcup_{k \in \llbracket 1, n \rrbracket} \{w'_k, k' \in \llbracket 1, d_k \rrbracket\} \right|$$

Soit  $M \in \mathcal{M}_{n,d}(\mathbb{N})$  la matrice du BOW de tous les documents. :

$$\forall i \in \llbracket 1, n \rrbracket, \forall j \in \llbracket 1, d \rrbracket, M[i, j] = \text{le nombre d'occurrence de } w_j \text{ dans le document } i$$

On obtient les propriétés suivantes :

**Propriété 3.1.0.3.** Soit  $M \in \mathcal{M}_{n,d}(\mathbb{N})$  : Alors, on peut définir la fonction d'appartenance d'un mot.

$$\forall k \in \llbracket 1, d \rrbracket, M[\cdot, w_k] : x \rightarrow \begin{cases} 1 & d_x[w_k] > 0 \\ 0 & \text{sinon} \end{cases}$$

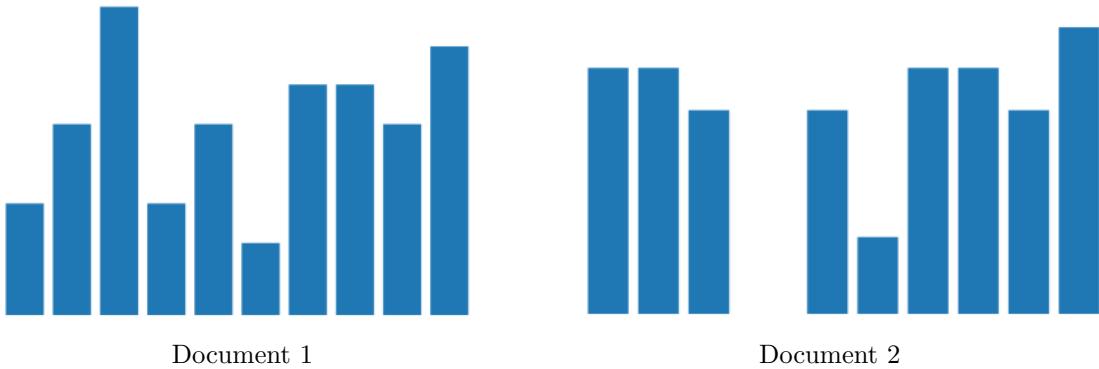


FIGURE 2 – Représentation de deux documents dans l'espace du vocabulaire.

## 3.2 TF-IDF

### 3.2.1 • TERM FREQUENCY

Une première méthode pour comparer les vecteurs ainsi obtenus est d'utiliser la similarité cosinus des deux vecteurs selon la formule suivante :

**Définition 3.2.1.1** (Similarité Cosinus - TF). Soit  $doc_A$  et  $doc_B$ , deux documents et de la taille du vocabulaire du corpus :

Alors

$$\rho(doc_A, doc_B) = \frac{\langle doc_A \cdot doc_B \rangle}{\|doc_A\| \|doc_B\|}$$

On remarque la propriété suivante :

**Propriété 3.2.1.2** (Positivité).  $\forall doc_A, doc_B, \rho(doc_A, doc_B) \in [0, 1]$

### 3.2.2 • TERM FREQUENCY, INVERSE DOCUMENT FREQUENCY

Cependant, pour plus de finesse dans l'analyse, on utilisera une version légèrement modifiée des documents. En effet, l'objectif est de caractériser facilement la différence entre nos documents d'un même corpus. Lorsqu'un document parle d'un thème, le vocabulaire du document et le champ lexical du thème en question sont fortement liés. Mais lorsque tous les documents parlent du même thème, la mesure simple de la similarité avec le champ lexical n'est pas suffisante pour apprécier la différence entre les deux documents. Ainsi, on pondère la fréquence des mots du document pour prendre en compte leur niveau de présence dans le corpus dont il est question. Pour cela, on introduit les définitions suivantes :

**Définition 3.2.2.1** (IDF). Soit  $w$  un mot dans le vocabulaire  $\mathcal{V}$  d'un corpus  $\mathcal{C}$  donné, on définit la fonction  $idf(word, document, corpus)$  de la façon suivante :

Soit

$$n_w = |\{M[x, w] > 0, \forall doc \in \mathcal{C}\}| = \text{le nombre de document où le mot apparaît}$$

$$idf(w, doc, \mathcal{C}) = \log\left(\frac{n}{n_w}\right)$$

Et on redéfinit donc les documents selon la pondération proposée :

**Définition 3.2.2.2.** Soit  $\tilde{doc}$  le nouveau document défini par :

$$\forall k \in [1, d], \tilde{doc}[w_k] = doc[w_k] * idf(w_k, doc, \mathcal{C})$$

On retrouve la définition proposée avec la similarité Cosinus, mais cette fois-ci, pondérée par les coefficients et on vérifie toujours la propriété de positivité.

### 3.3 ANALYSE

Après avoir utilisé l'algorithme de TF-IDF, on obtient par exemple les résultats suivants (limité aux 2 documents les plus proches pour des raisons de visibilité) :

Name	Doc1 - Values	Doc2 - Values
strategic framework	czech republic - 50%	czech republic - 24%
Report Strategy	ireland statement - 64%	ireland strategy - 49%
norway secondary	czech republic - 11%	lithuania ed - 9%
netherlands quality	netherlands quality - 24%	czech republic - 4%
netherlands quality	netherlands quality - 24%	denmark better - 13%
montenegro strategy	czech republic - 11%	hungary public - 9%
moldova national	moldova consolidated - 21%	czech republic - 14%
moldova education	moldova consolidated - 14%	georgia consolidated - 14%
moldova consolidated	moldova national - 21%	moldova education - 14%
lithuania ed	norway secondary - 9%	moldova consolidated - 8%
ireland strategy	Report Strategy - 49%	ireland statement - 46%
ireland statement	Report Strategy - 64%	ireland strategy - 46%
ireland national	ireland national - 23%	ireland department - 22%
ireland national	Report Strategy - 40%	ireland department - 40%
ireland department	ireland national - 40%	ireland statement - 35%
hungary public	czech republic - 16%	ireland strategy - 13%
georgia consolidated	albania education - 20%	moldova education - 14%
estonia youth	estonia higher - 23%	czech republic - 7%
estonia higher	estonia youth - 23%	czech republic - 13%
denmark better	czech republic - 18%	netherlands quality - 13%
czech republic	strategic framework - 50%	denmark better - 18%
czech republic	strategic framework - 24%	czech republic - 18%
croatia education	georgia consolidated - 9%	moldova national - 9%
bosnia and	ireland national - 6%	czech republic - 5%
armenia educational	moldova national - 10%	moldova consolidated - 9%
albania education	georgia consolidated - 20%	hungary public - 7%

FIGURE 3 – Résultats du TF-IDF sur le corpus de document européens

#### 3.3.1 • RÉSULTATS NOTABLES

On peut tout de même souligner quelques points intéressants :

Le TF-IDF permet de regrouper les documents par pays, bien qu'une simple lecture de leur nom suffise ici pour en distinguer la plupart, les documents issus d'un même pays ont bien plus de similarité entre eux avec les autres documents (avec l'exemple des documents sur l'Irlande de similarité supérieure à 40%).

De même, des similarités entre certains thèmes se dégagent : *Framework for Basic education* et *Strategy for*

*higher education* dégagent une similarité de 50%, un chiffre intéressant au vu de la différence entre l'école primaire et les études supérieures.

### 3.3.2 • DATASET ET FEEDBACK

Cependant, quelques remarques importantes :

- Chaque vecteur ne comporte que des composantes positives ou nulles, l'espace utile n'est donc que le demi-espace vectoriel des composantes positives puisqu'un mot ne peut pas apparaître un nombre négatif de fois. On ne modélise pas du tout la notion de contraire, les phrases "Je mange du chocolat" et "Je ne mange pas du chocolat" serait très similaire et pourtant de sens complètement opposé.
- Le TF-IDF ne prend pas en compte la sémantique des phrases. L'ordre des mots n'a pas d'importance dans la modélisation, mais il est pourtant primordial pour la bonne compréhension d'un texte. L'analyse est donc nécessairement partielle.
- L'algorithme souffre beaucoup du fléau de la dimension (*curse of dimensionality*) puisqu'on garde en mémoire, l'ensemble du vocabulaire des différents documents. Il est donc assez couteux de vérifier l'appartenance d'un mot lors du traitement d'un nouveau document...
- Enfin, il est difficile de justifier la similarité entre deux documents avec cette simple mesure. C'est à ce moment que nous avons rencontré une des plus grosses difficultés du projet : Le manque de donnée d'entraînement.

Dans cette partie et les suivantes, une des difficultés principales est de pouvoir évaluer la pertinence de nos modèles. Les documents fournis, au-delà de leur format peu manipulable, ne comportaient pas l'analyse de leurs propos. Il n'était pas envisageable de tous les lire nous même, par un manque crucial de connaissances pour en faire une analyse juste, mais également par manque de moyens pour en traiter autant (plusieurs dizaines de milliers de pages en cumulé).

C'est pourquoi, nous nous sommes efforcés d'utiliser des méthodes d'apprentissage non supervisé et restreint à des analyses simples pour évaluer les performances de nos méthodes.

Cependant, la méthode reste simple, intuitive et ouvre des possibilités pour faire des analyses rapides, des classifications légères et surtout affiner d'autres modèles pour reconnaître les mots les plus importants d'un texte.

## 4 GLOVE

---

### 4.1 INTRODUCTION

---

Nous avons vu que le principal défaut du Doc2Vec était qu'il représentait tout un texte par un seul vecteur. Ainsi, il est normal que nous perdions beaucoup d'informations à travers cette méthode de représentation. Dans la continuité de cette démarche de représentation vectorielle, il est naturel de s'intéresser à la méthode de représentation GloVe, où chacun des vecteurs est reflété par un vecteur !

En effet, si nous pouvions représenter chacun des mots par des vecteurs, et faire en sorte que des mots de même sens aient des vecteurs similaires, nous pourrions essayer de dégager différents axes (thèmes) dans nos données.

L'idée principale de GloVe est qu'il prend en compte le contexte des mots. Pour un mot  $m$  donné, l'algorithme permet de créer des vecteurs de dimension arbitraire en tenant compte d'une fenêtre de contexte dans laquelle  $m$  apparaît. La dimension du vecteur sera alors égale à la taille de la fenêtre considérée.

Pour comprendre comment l'algorithme fonctionne, nous devons d'abord définir la notion de cooccurrence.

**Définition 4.1.0.1** (Cooccurrence). Soit  $L$  une liste de mots. Soient deux mots  $m_1$  et  $m_2$  de  $L$ , et  $d$  un entier naturel.

La cooccurrence de  $m_1$  et de  $m_2$  est l'entier  $n$  qui représente le nombre de fois où les mots  $m_1$  et  $m_2$  sont apparus à moins de  $d$  mots l'un de l'autre.

**Exemple 4.1.0.2** (Cooccurrence). Prenons la phrase : "J'ai mangé des légumes à midi, le soir, j'ai mangé des frites.". On a alors, pour  $d = 1$ ,

$$\text{Occ}(\text{"mangé"}, \text{"des"}) = 2$$

$$\text{Occ}(\text{"mangé"}, \text{"légumes"}) = 0$$

### 4.2 FONCTIONNEMENT DE GLOVE

---

#### 4.2.1 • NOTATIONS

Soit  $L$  la liste de mots représentant notre texte (ou corpus de textes) sur lequel on veut entraîner notre modèle. Ainsi, on dispose d'un ordre des mots :  $L = [m_1, m_2, \dots, m_N]$ . La première étape est de construire une matrice de cooccurrence  $X \in \mathbb{R}^{N \times N}$ , telle que  $X_{i,j}$  décrit la cooccurrence entre  $m_i$  et  $m_j$ . Cela nous permet de déterminer la forme idéale pour nos vecteurs.

Notons  $X_i = \sum_k X_{i,k}$  le nombre d'occurrences total de tous les mots du texte et  $P_{i,j} = P(i|j) = X_{i,j}/X_i$  la probabilité que le mot  $m_j$  apparaisse en présence du mot  $m_i$  dans le contexte. On introduit également une notion intéressante : les vecteurs de contexte.

**Définition 4.2.1.1** (Vecteurs de contexte). Pour chaque mot  $m_i$  du texte, on lui associera essentiellement deux vecteurs :  $w_i$  qui est le vecteur de représentation global, et  $\tilde{w}_i$  qui est le vecteur de représentation du mot, en tenant seulement compte des mots qui se situent avant et après de  $m_i$  dans la fenêtre de contexte donnée.

#### 4.2.2 • FONCTIONNEMENT

L'observation principale est que le sens des mots peut être encodé par son contexte, et plus particulièrement par le ratio entre deux probabilités conditionnelles.

Ainsi, il semblerait qu'un encodage des vecteurs  $w_i$  tel que :

- $w_i \cdot \tilde{w}_k = \log(P(i|j))$
- $(w_i - w_j) \cdot \tilde{w}_k = \log \frac{P(i|k)}{P(j|k)}$  pour garantir que l'on puisse manipuler les composantes de sens des vecteurs de façon linéaire. Par exemple :  $w_{king} - w_{queen}$  devrait signifier "masculin".

soit souhaitable pour avoir notre représentation vectorielle de nos mots en tenant compte des cooccurrences.

Ainsi, nous arrivons à l'expression suivante :

$$w_i \cdot \tilde{w}_k = \log(P(i|k)) = \log(X_{i,k}) - \log(X_i) \quad (1)$$

Si bien que nous pouvons écrire :

$$w_i \cdot \tilde{w}_k + b_i + \tilde{b}_k = \log(X_{i,k}) \quad (2)$$

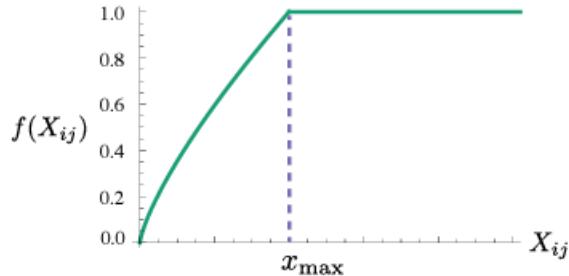
Nous avons rajouté des termes de biais  $b_i$  pour comprendre  $\log(X_i)$  et nous avons rendu la relation symétrique avec le terme  $\tilde{b}_k$ . Nous avons ainsi atteint notre premier objectif : **trouver une relation entre les vecteurs de mots avec leur cooccurrence**.

Ainsi, le modèle GloVe tente de se rapprocher le plus possible de l'équation (2) pour tous les vecteurs de ses mots. Il essaie donc de minimiser la fonction de perte suivante :

**Propriété 4.2.2.1** (Fonction de perte de GloVe). Soient  $N$  le nombre de mots total du texte,  $d$  entier naturel qui représente la taille de la fenêtre de contexte,  $w_i, \tilde{w}_i \in \mathbb{R}^d$  vecteurs de représentation du mot  $m_i$ . La fonction de perte du modèle GloVe s'écrit comme suit :

$$J = \sum_{i,j}^N f(X_{i,j})(w_i \cdot \tilde{w}_k + b_i + \tilde{b}_k - \log(X_{i,k}))^2 \quad (3)$$

Avec  $f$  une fonction de poids qui permet d'accorder de l'importance aux mots qui coocurrent le plus.

FIGURE 4 – Représentation graphique d'une fonction  $f$  potentielle

#### 4.2.3 • MINIMISATION DE LA FONCTION DE PERTE : MÉTHODE DE DESCENTE DE GRADIENT

La méthode de la descente de gradient permet de trouver un point de minimum global.

**Propriété 4.2.3.1** (Descente de gradient). *Soit  $J : \mathbb{R}^d \rightarrow \mathbb{R}$  et  $\mu > 0$  une fonction de perte différentiable. L'algorithme de la descente de gradient à pas fixe consiste à construire une suite  $(u_n)_{n \in \mathbb{N}}$  telle que :*

$$u_{n+1} = u_n - \mu J'(u^n) \quad (4)$$

*Pour une fonction  $J$  strictement convexe, différentiable et infinie à l'infinie, telle que  $J'$  est Lipschitzien, pour  $\mu$  suffisamment petit, la suite  $(u_n)$  converge vers le minimiseur de  $J$ .*

Cependant, la fonction de perte de GloVe n'est pas strictement convexe (à cause de la fonction de poids  $f$ ). Pour assurer la convergence de l'algorithme de descente de gradient, GloVe initialise les vecteurs  $w_i$  et  $\tilde{w}_i$  de façon aléatoire, de sorte que l'algorithme explore différentes parties de l'espace pour ne pas rester bloqué dans un minimum local. De plus, l'algorithme *Adam* pour ajuster le pas de la descente à chaque itération.

### 4.3 APPLICATIONS

---

#### 4.3.1 • QUELQUES EXEMPLES SIMPLES

Le dictionnaire GloVe de dimension 300 dispose de propriétés très intéressantes. En effet, il dispose de résultats expérimentaux très satisfaisants.

Il nous est par exemple très facile d'obtenir des synonymes ou des mots de sens proches d'un autre mot. En effet, en utilisant la similarité cosinus que nous avons déjà évoquée, nous pouvons déterminer pour un mot donné les vecteurs qui lui sont le plus similaires. Ce faisant, nous arrivons à des résultats satisfaisants pour le mot "frog".

**Exemple 4.3.1.1** (Mots similaires à "frog"). *Les mots les plus proches déterminés par similarité cosinus sont :*

1. frogs
2. toad
3. litoria
4. leptodactylidae



FIGURE 5 – À gauche, une image d'une litoria, à droite, celle d'un leptodactylus selon Wikipédia

**Propriété 4.3.1.2** (Mots et synonymes). *Deux mots de sens similaires ont une grande similarité cosinus (proche de 1).*

De plus, nous pouvons remarquer que le modèle GloVe permet de manipuler les composantes de sens de façon linéaire entre les différents vecteurs. Il s'agissait en fait d'un de nos objectifs de base lors de la construction de la fonction de perte.

**Propriété 4.3.1.3** (Vecteur de sens). *La nuance entre deux mots de sens similaire peut être capturée en une différence de vecteurs.*

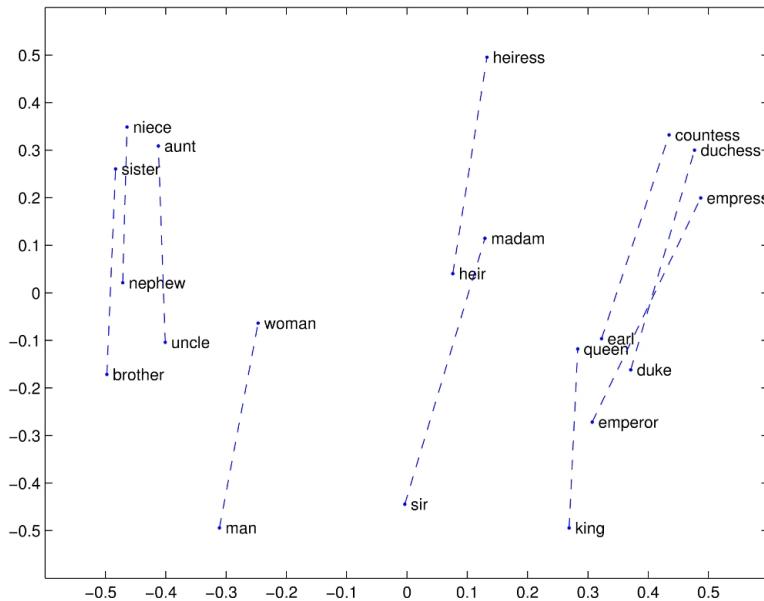


FIGURE 6 – Vecteur de sens homme/femme. Tiré de *GloVe : Global Vectors for Word Representation*. Jeffrey Pennington, Richard Socher, and Christopher D. Manning, 2014.

Dans ce graphique, des vecteurs de mots ont pu être représentés sur un graphique en 2 dimensions (les

techniques de réduction de dimension seront expliquées dans la partie suivante). On peut remarquer que la composante de sens *homme* → *femme* est environ la même entre tous les couples de mots (homme, femme).

#### 4.3.2 • APPLICATION AU CORPUS DE TEXTES

Nous avons donc appliqué le modèle de GloVe pré-entraîné à notre corpus de textes. Nous avons pu ainsi construire un dictionnaire qui associe à chaque mot de notre corpus un vecteur de dimension 300. Ainsi, nous avons pu plonger notre texte en vecteurs qui ont la propriété particulière d'être similaires lorsque les mots qu'ils représentent ont des sens similaires.

Néanmoins, même si une machine peut comprendre les relations entre vecteurs de dimension 300, il nous est préférable de réduire la dimension de ces vecteurs pour avoir quelque chose de plus visuel. L'objectif est donc de les réduire en dimension 2 ou 3 pour pouvoir projeter les mots dans un graphique que l'humain pourrait comprendre.

Il existe différents algorithmes de réduction de dimension tels que la *Principal Component Analysis*, la *Single Value Decomposition*, ou encore la *t-Distributed Stochastic Neighbor Embedding*. C'est cette dernière méthode que nous avons décidé d'utiliser. En effet, la *t-SNE* est une technique de réduction de dimension souvent utilisée en *NLP* qui a la particularité de conserver la structure locale des différents vecteurs, leurs différents voisins.

**Propriété 4.3.2.1** (Conservation de la structure locale). *L'algorithme de réduction de dimension t-Distributed Stochastic Neighbor Embedding permet de maintenir les distances des points qui sont proches en haute dimension, et de les rapprocher dans l'espace à dimension réduite. À l'inverse, l'algorithme écarte les points qui sont moins similaires (c.-à-d. qui sont éloignés en dimension élevée).*

Ainsi, en réduisant la dimension des différents vecteurs en utilisant un algorithme de *t-SNE*, on se garantit de garder la propriété 4.3.1.2 en basse dimension ! Ainsi, nous pouvons exploiter nos vecteurs sur un graphique de dimension 2 ou 3. Nous allons à présent essayer de déterminer les axes principaux d'un texte à partir des mots qu'il emploie le plus souvent. Nous allons donc introduire différentes méthodes de *clustering* pour répondre à ce problème.

## 5

# CLUSTERING

---

### 5.1 INTRODUCTION

---

Le *clustering* de données est une technique d'analyse de données largement utilisée en NLP pour regrouper des textes similaires en ensembles appelés *clusters*. L'objectif est d'utiliser des mesures d'un élément, des caractéristiques, pour déterminer sa classe d'appartenance (son *cluster* donc). Plus particulièrement en NLP, le *clustering* est utilisé pour de la classification de textes, de mots, de document... Ici, l'objectif est de déterminer les thèmes d'un document en supposant que les mots d'un même thème présentent des caractéristiques similaires. En particulier donc, à partir du modèle de représentation GloVe, donnant les caractéristiques d'un mot d'un document, de déterminer les thèmes dudit document.

### 5.2 ALGORITHMES

---

Pour cela, nous avons envisagé plusieurs techniques, en particulier, plusieurs algorithmes de *clustering*.

#### 5.2.1 • K-MEANS

K-means est un des algorithmes de *clustering* les plus connus. Il permet de subdiviser un jeu de données en  $k$  (paramètre prédéfini en amont) classes. Voilà un résumé succinct de son principe :

**Définition 5.2.1.1** (K-Means). *On définit l'algorithme K-Means de la façon suivante :*

1. *Initialisation de l'algorithme avec les centres  $\mu_1, \mu_2, \dots, \mu_k$  au hasard.*
2. *Assigner à chaque élément  $x_i$  le centre le plus proche, c'est-à-dire, trouver le  $j$  celui qui minimise  $\|x_i - \mu_j\|^2$ .*
3. *Mettre à jour les centres avec le calcul du barycentre des éléments qui lui sont assignés*
4. *Répéter les étapes 2 et 3 jusqu'à la convergence de l'algorithme*

**Propriété 5.2.1.2** (K-Means). *On note les propriétés suivantes pour K-Means*

- *Les clusters sont à union disjointe, pour des thèmes en NLP, cela veut donc dire que chaque mot appartient à unique thème.*
- *K-Means est hautement sensible à l'initialisation de départ, il est primordial d'utiliser K-Means++ pour optimiser l'initialisation.*
- *Les clusters sont présupposés sphériques et isotropes, signifiant que toutes les variances sont égales.*
- *La convergence de l'algorithme n'est pas garantie, il est difficile de trouver le minimum global.*
- *La métrique de distance utilisée n'est pas nécessairement optimale pour le corpus de données considéré.*
- *Les clusters formés sont des partitions convexes de l'ensemble possible des éléments, mais ce n'est pas toujours le cas dans la réalité.*

#### 5.2.2 • DBSCAN

**Définition 5.2.2.1** (DBscan). *DBSCAN (Density-Based Spatial clustering of Applications with Noise) est un algorithme de clustering basé sur la notion de densité. Il regroupe les points proches les uns des autres et marque les points isolés comme du bruit.*

*Soit  $X$  un ensemble de  $n$  points dans  $\mathbb{R}^d$  et, soit  $\epsilon$  et MinPts deux paramètres de l'algorithme. On définit le voisinage d'un point  $x$  comme l'ensemble des points à la distance  $\epsilon$  de  $x$  tel que :*

$$N_\epsilon(x) = \{y \in X \mid \text{dist}(x, y) \leq \epsilon\}$$

*Un point  $x$  est considéré comme un point cœur si au moins MinPts points sont dans son voisinage, et un point de frontière s'il a moins de MinPts points dans son voisinage, mais qu'il est dans le voisinage d'un point cœur. Tous les autres points sont considérés comme du bruit.*

*L'algorithme procède en sélectionnant un nombre arbitraire de points, il vérifie s'ils sont des coeurs et il explore en procédant à la même analyse avec les points dans les voisinages. Une fois que tous les points coeurs sont visités, les points bordures restants sont marqués comme du bruit.*

**Propriété 5.2.2.2** (DBscan). *L'algorithme possède les propriétés suivantes :*

- Contrairement à K-means, le nombre de clusters est donné par l'algorithme avec le nombre de points coeurs et peut donc déterminer des clusters peu importe leurs formes.
- L'algorithme est moins sensible au bruit, il écarte ces points de l'étude.
- L'algorithme est très fortement sensible au choix des paramètres de départ ! Il faut connaître quelques éléments sur la structure à analyser.
- L'algorithme tend à minimiser le nombre de clusters. En particulier, il produit de moins bons résultats lorsque la densité des clusters est variable.

**Définition 5.2.2.3** (Optics). *Optics est un autre algorithme de clustering basé sur la densité. Il a un fonctionnement similaire à DBscan aux différences suivantes :*

- Optics est plus couteux en temps de calcul que DBscan ( $o(n^2) > o(n * \log(n))$ )
- Les points qualifiés de "bruit" pas DBscan sont simplement considérés comme "très peu accessibles" par Optics qui qualifie les points par un taux d'accessibilité à la place.
- Optics ne nécessite que le paramètre de nombre minimum de points. Il calcule pour tout un intervalle de valeurs de  $\epsilon$  possible en passant par un graphe de connectivité.
- Optics renvoie un cluster hiérarchique plus riche donc que le cluster de DBscan.

Pour une première approche naïve, nous avons utilisé Optics avant DBscan mais nous avons finalement opté pour l'utilisation de K-Means qui produit de meilleurs résultats (vois ci-dessous). Nous utilisons le modèle GloVe qui décrit avec finesse chaque mot comme un point comportant 300 coordonnées et nous cherchons à obtenir des classes d'appartenance pour les mots d'un même document.

## 5.3 RÉSULTATS & ANALYSE

Dans un souci de représentation des résultats, il est souvent envisagé de réduire la dimension de nos éléments jusqu'à obtenir 2 ou 3 dimensions et mieux visualiser. C'est ce que nous avons effectué sur les résultats des modèles précédent à l'aide de *PCA* comme expliqué précédemment. Ainsi, nous obtenons ce type de figure :

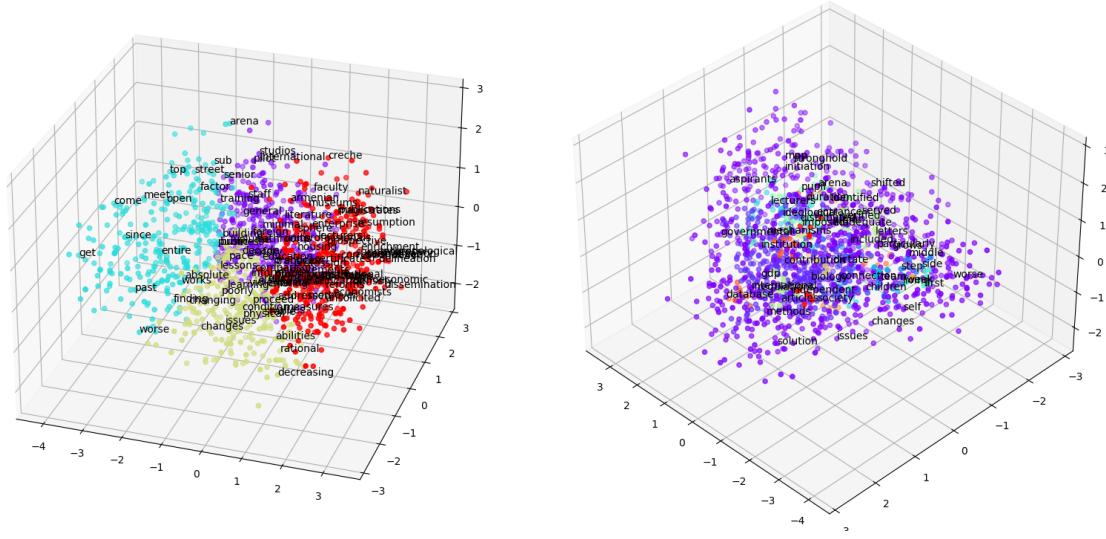


FIGURE 7 – À gauche, K-Means ( $k=4$ ), à droite, DBscan ( $\epsilon = 0.3, n_{min} = 5$ )

On remarque que K-Means traite bien plus de résultats, on regarde donc plus en détails les *clusters* proposés avec un renfort de la méthode de TF-IDF pour sélectionner les éléments les plus importants : On trie les mots de chaque *cluster* selon leur importance dans la modélisation TF-IDF du document dans le corpus considéré. On obtient par exemple les résultats suivants pour le document ci-contre :

Top Words :	<i>cluster 1</i>	<i>cluster 2</i>	<i>cluster 3</i>	<i>cluster 4</i>
0	needing	doc	quarter	armenia
1	decreased	program	mid	roa
2	approval	programs	fourth	elaboration
3	textbooks	secondary	school	upbringing
4	attachment	expansion	field	cadres
5	acts	organization	tv	attestation
6	loaded	republican	performances	diaspora
7	imperfect	military	children	armenian
8	suffered	boarding	graduate	pedagogical
9	destroyed	formation	pre	utilization
10	decline	preschool	radio	normative
11	lack	republic	elementary	rationalization
12	incomplete	laws	turning	exemplary
13	definition	ph	contest	schooling
14	grades	broadcasting	typical	yearly

FIGURE 8 – *Clusters* trouvés par K-Means

On peut donner comme interprétations les thèmes suivants :

- *Cluster 1* : Performance Académiques et Défis
- *Cluster 2* : Educations et Institutions
- *Cluster 3* : Développement
- *Cluster 4* : Culture et Education arménienne

Cependant, on remarque aisément que la définition des thèmes n'est pas aisée. Bien que les thèmes proposés puissent convenir, il est difficile de dire que ce sont les seuls. La méthode proposée, bien qu'encourageante pour définir des thèmes, manque encore beaucoup de précision. On pourrait implémenter une méthode qui permet de ne conserver seulement les noms et verbes (et non plus adjectifs), puisque l'on remarque ici beaucoup d'adjectifs qui n'apportent pas vraiment de sens au *cluster*.

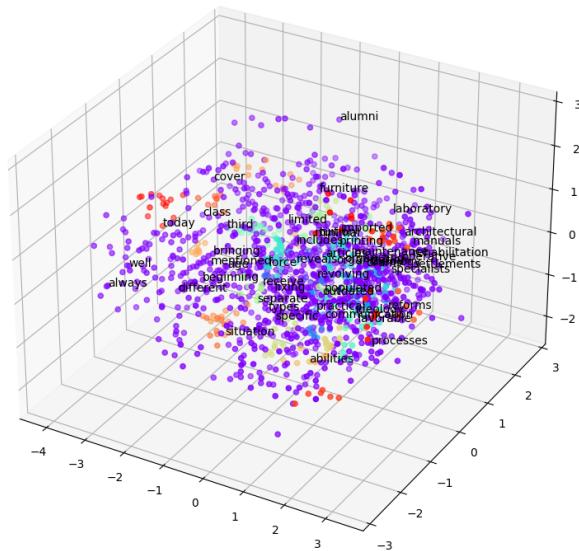


FIGURE 9 – Clustering réalisé par Optics  $n_{min} = 5$

## 6

# TOPIC MODELLING ET REGEX

### 6.1 INTRODUCTION

Le *Topic Modeling* est un type d'apprentissage statistique non supervisé qui identifie des *clusters* ou groupe de mots similaires dans un texte.

Cette méthode utilise les structures sémantiques dans le texte pour comprendre les données non structurées sans label prédéfini ou jeu de données d'entraînement, elle analyse les documents pour identifier les thèmes communs et fourni un *cluster* adéquat. En l'occurrence, dans notre cas, elle nous permettra de pouvoir extraire les thèmes prépondérants dans notre corpus de texte.

Parmi les différentes techniques de *Topic Modeling*, on en retrouve quatre qui sont très utilisées :

- **LDA (Latent Dirichlet Allocation)** : Le modèle LDA est un modèle probabiliste génératif et itératif qui suppose que chaque document dans un corpus de texte (dans notre cas, les plans d'éducation nationale) est un mélange de différents thèmes, et chaque thème est une distribution des mots. LDA utilise une distribution de **Dirichlet** pour initialiser son modèle, et modélise la distribution des mots et des thèmes simultanément.
- **pLSA (Probabilistic Latent Semantic Analysis)** : La pLSA est un modèle génératif qui suppose que chaque document est un mélange de différents thèmes, mais contrairement à LDA, qui utilise une distribution de **Dirichlet** pour modéliser la distribution des thèmes, la pLSA apprend la distribution thème-mot et la distribution de chaque thème pour chaque document séparément en utilisant un algorithme EM (espérance-maximisation).
- **HPD (Hierarchical Dirichlet Process)** : Ce modèle est une extension de la LDA qui autorise le document à avoir un nombre non borné de thèmes. Le modèle suppose que les thèmes du document suivent une structure hiérarchique et déduit le nombre de thèmes et la distribution thème-mot automatiquement.
- **CTM (Correlated Topic Models)** : Ce modèle est également une extension de la LDA qui autorise les thèmes à avoir une corrélation. Le modèle suppose que chaque thème est généré suivant une distribution Gaussienne et la corrélation entre les thèmes est obtenu via la matrice de covariance.

### 6.2 ALLOCATION DE DIRICHLET LATENTE (LDA)

#### 6.2.1 • NOTATIONS ET DÉFINITIONS

Nous définissons les termes suivants :

**Définition 6.2.1.1** (Corpus). *Collection de  $M$  documents  $D = \{w_1,..w_M\}$  où  $w_d$  est le  $d$ -ième document de la collection.*

**Définition 6.2.1.2** (Document). *Séquence de  $N$  mots dénotée par  $w_d = \{w_{d,1},..w_{d,N}\}$  où  $w_{d,n}$  est le  $n$ -ième mot de la séquence du  $d$ -ième document du corpus.*

**Définition 6.2.1.3** (Mot). *Unité de base de nos données discrétisées. C'est un élément au sein d'un vocabulaire  $V$  indexé par  $\{1, ..., L\}$ .*

Le modèle de la LDA considère que le corpus  $D$  de documents est constitué d'un nombre fixé  $K$  de thèmes. De plus, ce corpus est caractérisé par deux distributions de probabilité.

**Propriété 6.2.1.4.** Pour chaque thème  $k \in \{1, \dots, K\}$  est associé une distribution de probabilités  $\beta_k$  sur les mots du vocabulaire. Chaque probabilité est celle qu'un mot soit dans le thème  $k$  :  $\beta_k = \{p(w|k)\}_{w \in V}$

**Propriété 6.2.1.5.** Pour chaque document  $w_d$ ,  $d \in \{1, \dots, M\}$ , est associé une distribution de probabilités  $\theta_d$  sur les thèmes. Chaque probabilité représente la proportion du thème dans le document  $w_d$  :  $\theta_d = \{p(k|w_d)\}_{k=1, \dots, K}$ .

La LDA est un modèle probabiliste génératif de corpus de texte. Il se base sur une description selon laquelle les données, à savoir nos documents, ont été générées. Cette description prend en compte des variables latentes qui permettent de créer les données. Par exemple, les variables  $\beta_k$  et  $\theta_d$  sont latentes. Notre corpus  $D$  sera supposé créé selon le processus génératif de la LDA. L'idée est alors de bien comprendre le processus afin d'essayer de retrouver les variables latentes à partir du corpus  $D$  : **rétro-ingénierie**.

### 6.2.2 • PROCESSUS GÉNÉRATIF

Dans cette section, nous allons décrire le modèle de génération de documents de la LDA et selon lequel on supposera que nos données ont été générées.

La première chose à faire pour générer un document consiste à choisir le nombre de mots  $N$  qu'il contient. Pour générer un document  $w_d$ , le modèle commence par fixer le nombre de mots  $N$  au sein du document. Ce nombre est fixé suivant une loi de poisson de paramètre  $\lambda$ .

$$N \sim \text{Poisson}(\lambda)$$

Le paramètre  $\lambda$  est aussi une variable latente du modèle, mais nous ne chercherons pas à retrouver sa valeur dans notre démarche de rétro-ingénierie. En effet, nous rappelons que l'objectif est l'extraction des  $K$  thèmes au sein des documents exprimés par les distributions  $\beta_k$ ,  $\theta_d$  ainsi que par d'autres variables que nous définirons par la suite.

Il faudra ensuite peupler  $N$  positions correspondant aux mots  $w_{d,n}$  pour  $n = 1, \dots, N$  de notre document. Pour chaque position  $n$  à peupler, on commence par choisir un thème  $z_{d,n}$  suivant la distribution probabiliste  $\theta_d$ .

$$z_{d,n} \sim \text{Multinomial}(\theta_d)$$

Ensuite,  $w_n$  est pris selon la distribution  $\beta_{z_n}$  du thème  $z_n$ . On rappelle que les thèmes sont définis comme une distribution de probabilité sur les mots du vocabulaire.

$$w_{d,n} \sim \text{Multinomial}(\beta_{z_{d,n}})$$

Le modèle de LDA fait la supposition que les distributions de  $\beta_{1:K}$  et  $\theta_{1:d}$  sont de Dirichlet.

**Définition 6.2.2.1** (Loi de Dirichlet). Soit  $x = (x_1, \dots, x_n)$  une variable aléatoire vérifiant  $\sum_{i=1}^K x_i = 1$ ,  $x_i \in [0, 1] \forall i \in \{1, \dots, K\}$ .

La loi de Dirichlet d'ordre  $K \geq 2$  et de paramètres  $\alpha_1, \dots, \alpha_K > 0$  possède pour densité de probabilité :

$$p(x) = \frac{1}{B(\alpha)} \prod_{i=1}^K x_i^{\alpha_i - 1} \quad (5)$$

avec  $\alpha = (\alpha_1, \dots, \alpha_K)$  et  $B$  étant la fonction bêta qui sert de constante de normalisation.

Illustrons la loi de Dirichlet à l'aide d'un exemple à l'ordre 2. Supposons que l'on dispose d'une pièce de monnaie dont on souhaiterait connaître la probabilité  $x_1$  d'obtenir pile et  $x_2$  d'obtenir face. Pour obtenir ces probabilités, la première intuition serait de lancer la pièce plusieurs fois et de noter le ratio piles-faces.

Notons :

- $N$  : le nombre total de lancers
- $N_p$  : le nombre de piles obtenu
- $N_f$  : le nombre de faces obtenu

On a alors

$$\lim_{N \rightarrow \infty} \frac{N_p}{N} = x_1$$

Or pour un nombre fini de lancers, on ne peut pas déterminer  $x = (x_1, x_2)$  avec précision. On parle alors d'une variable aléatoire  $x$ . La distribution de Dirichlet permet de modéliser sa densité de probabilité.

En effet, en posant  $\alpha_1 = N_p + 1$  et  $\alpha_2 = N_f + 1$  on peut faire la supposition que  $x = (x_1, x_2) \sim Dir(\alpha_1, \alpha_2)$ . La densité de probabilité est maximale en  $(\frac{N_p}{N}, \frac{N_f}{N})$ . Autrement dit, l'estimation "la plus probable" de  $x$  est  $(\frac{N_p}{N}, \frac{N_f}{N})$ . Ceci concorde avec l'intuition qui selon laquelle on estime  $x_0$  par  $\frac{N_p}{N}$ .

Dans le cas général, plus  $\alpha_i$  sera plus élevée et plus  $x_i$  aura de "poids" par rapport aux autres composantes de  $x$ . Si les  $\alpha_i$  sont égaux, la distribution est symétrique. Si  $\alpha_i < 1$ , il peut être considéré comme un anti-poids qui repousse  $x_i$  vers les extrêmes, alors que lorsqu'il est élevé, il attire  $x_i$  vers une valeur centrale (centrale dans le sens où tous les points sont concentrés autour d'elle). Si  $\alpha_1 = \dots = \alpha_K = 1$ , alors les points sont uniformément répartis.

Cela peut être vu sur les graphiques ci-dessous, où l'on a représenté des distributions de Dirichlet d'ordre 3 pour des valeurs différentes de  $\alpha$ .

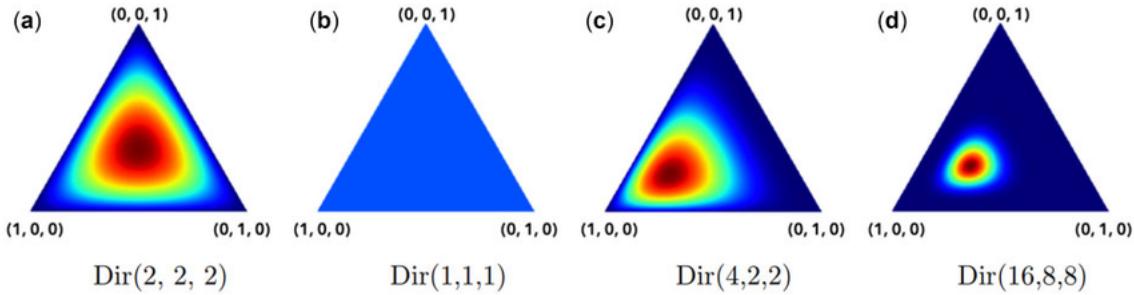


FIGURE 10 – Distributions de Dirichlet d'ordre 3

De la même façon que nous avons associé la distribution de probabilités pour la pièce de monnaie à une loi de Dirichlet, le modèle suppose que les  $\beta_k$  et  $\theta_d$  sont respectivement des lois de Dirichlet de paramètre  $\eta$  et  $\alpha$ .

Nous avons désormais défini les éléments nécessaires pour décrire le modèle de génération de la LDA. Elle suppose que chaque document  $w_d$  du corpus  $D$  est généré de la façon suivante :

1. Choisir  $N \sim Poisson(\lambda)$
2. Choisir  $\theta_d \sim Dir(\alpha)$
3. Pour chacun des  $N$  mots de  $d$  faire :
  - (a) Choisir  $z_{d,n} \sim Multinomial(\theta_d)$

- (b) Choisir  $\beta_{z_{d,n}} \sim Dir(\eta)$
- (c) Choisir  $w_{d,n} \sim Multinomial(\beta_{z_{d,n}})$

Pour résumer le tout, la figure ci-dessous représente de façon graphique le modèle de la LDA. Avec :

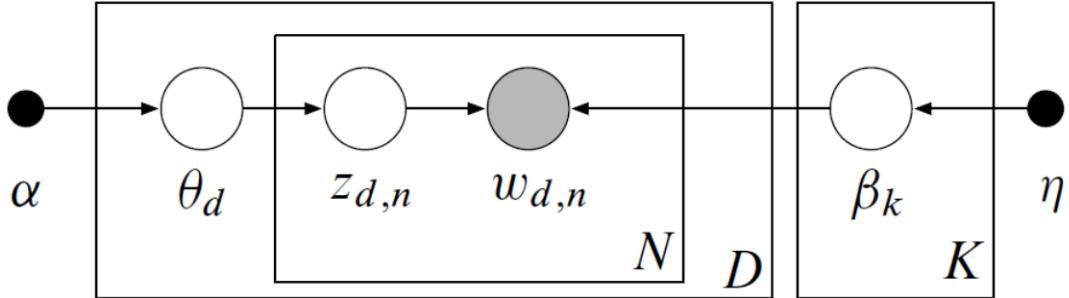


FIGURE 11 – Modèle graphique de la LDA

- $\alpha$  et  $\eta$  les paramètres des distributions de Dirichlet  $\theta_d$  et  $\beta_k$
- $\theta_d$  la distribution des thèmes pour le document  $d$
- $\beta_k$  la distribution des mots pour le thème  $k$
- $z_{d,n}$  le thème du  $n$ -ième mot du document  $d$
- $w_{d,n}$  le  $n$ -ième mot du document  $d$

### 6.2.3 • RETROUVER LES VARIABLES LATENTES

Maintenant que nous avons décrit le fonctionnement du modèle génératif, nous allons voir comment retrouver les variables latentes (rétro-ingénierie) connaissant le corpus  $D$  supposé généré à partir du modèle. Nous sommes intéressés par la connaissance de trois variables latentes :

- L'ensemble  $z$  des thèmes des mots au sein des documents :  $z = \{z_{d,n}\}$
- L'ensemble  $\beta$  de nos distributions  $\beta_k$  :  $\beta = \{\beta_1, \dots, \beta_K\}$
- L'ensemble  $\theta$  de nos distributions  $\theta_k$  :  $\theta = \{\theta_1, \dots, \theta_K\}$

Nous allons rassembler ces trois variables aléatoires en une nouvelle variable  $X = (\theta, \beta, z)$  dont on cherchera la valeur la plus probable connaissant  $D$ . Autrement dit, cela revient à un programme de maximisation :

$$\max_X \left[ p(X|D) = \frac{p(X, D)}{p(D)} \right]$$

Il est important de noter que les paramètres  $\alpha$  et  $\eta$  restent fixés (c'est à l'utilisateur du modèle de les choisir).

Le numérateur est calculable à partir de la formule :

$$\begin{aligned} p(X, D) &= p(\theta, \beta, z, D) = p(\theta)p(\beta)p(z|\theta)p(D|z, \beta) \\ &= \prod_{k=1}^K p(\beta_k) \prod_{d=1}^M p(\theta_d) \prod_{n=1}^N p(z_{d,n}|\theta_d)p(w_{d,n}|z_{d,n}, \beta) \end{aligned}$$

Aussi, en intégrant par rapport à  $\theta$ ,  $\beta$  et en sommant sur les valeurs possibles de  $z$ , on peut retrouver  $p(D)$  par :

$$p(D) = \int_{\theta} \int_{\beta} p(\theta)p(\beta) \sum_z p(z|\theta)p(D|z, \beta)$$

Dans la pratique,  $p(X, D)$  est calculable. Cependant, l'intégrale donnant  $p(D)$  n'est pas toujours calculable, surtout lorsque le nombre  $M$  de documents est très élevé. En effet, il va falloir intégrer sur  $\theta_1, \dots, \theta_M$ . Nous n'avons donc pas accès à la distribution  $p(X|D)$ . Heureusement, des méthodes d'approximation de la distribution de  $p(X|D)$  existent. Notre code informatique se base sur une technique appelée l'inférence variationnelle (VI). Une fois que nous avons approché  $p(X|D)$  par une autre distribution  $q(X)$ , il suffit de choisir  $X$  correspondant à la valeur maximale de  $q(X)$ .

#### 6.2.4 • INFÉRENCE VARIATIONNELLE (VI)

L'idée est d'essayer de résoudre un problème d'optimisation sur une classe de distributions traitables  $Q$  (des lois normales par exemple) afin de trouver un  $q(X) \in Q$  qui ressemble le plus à  $p(X|D)$ . Dans notre cas, la divergence K-L servira de mesure de dissimilarité entre deux distributions de probabilités.

**Définition 6.2.4.1** (Divergence K-L). *Soient  $p, q$  deux distributions de probabilités discrètes, la divergence K-L de  $p$  par rapport à  $q$  est définie par :*

$$KL(q||p) = E_{X \sim q(X)}(\log(\frac{q(X)}{p(X)})) = \sum_x q(x) \log(\frac{q(x)}{p(x)})$$

*La distribution K-L est généralisable pour des distributions continues par :*

$$KL(p||q) = \int_x q(x) \log(\frac{q(x)}{p(x)})$$

La divergence KL a les propriétés suivantes particulièrement utiles dans notre contexte :

- $KL(q||p) \geq 0 \quad \forall p, q$
- $KL(q||p) = 0 \iff q = p$

On cherche donc  $q(X)$  qui minimise  $KL(q(X)||p(X|D))$ .

$$\begin{aligned} KL(q(X)||p(X|D)) &= \int_x q(x) \log(\frac{q(x)}{p(x|D)}) \\ &= \int_x q(x) \log(\frac{q(x)p(D)}{p(x, D)}) \\ &= \int_x q(x) \log(\frac{q(x)}{p(x, D)}) + \int_x q(x) \log(p(D)) \\ &= -E_{X \sim q(X)}(\log(\frac{q(x)}{p(x, D)})) + \log(p(D)) \end{aligned}$$

Étant donné que  $\log(p(D))$  est une constante ne dépendant pas de  $q(X)$ , minimiser  $KL(q(X)||p(X|D))$  revient à maximiser  $L(q) = E_{X \sim q(X)}(\log(\frac{q(x)}{p(x, D)}))$ . Nous avons donc réussi à contourner le problème que levait la non-calculabilité de  $p(D)$ . Puisque  $KL(q(X)||p(X|D)) \geq 0$ ,  $L(q) \leq \log(p(D))$  et  $q(X) = p(X) \iff L(q) = \log(p(D))$ .

### 6.3 APPLICATION À NOTRE JEU DE DONNÉES

Pour la partie pratique, nous avons utilisé le modèle pré-implémenté `gensim.models.ldamodel.LdaModel` de la librairie python Gensim. Ce modèle utilise l'inférence variationnelle pour estimer les paramètres du modèle d'Allocation de Dirichlet Latente.

L'approche d'inférence variationnelle est utilisée, car elle permet une estimation efficace et évolutive des paramètres du modèle LDA, ce qui peut être difficile en utilisant d'autres techniques d'inférence. VI est également rapide en termes de calcul, ce qui en fait un choix populaire pour les applications d'analyse de texte à grande échelle.

### 6.3.1 • PREMIERS TESTS

Après avoir compris comment fonctionnait l'algorithme et surtout le genre de résultat qu'il pourrait nous fournir, nous nous sommes penchés sur son application à notre jeu de données, comme ce dernier est d'une taille assez conséquente, nous avons commencé par une application sur un seul texte pour vérifier la précision du modèle.

Voici les détails du premier test que nous avons effectué :

- **Document utilisé :** *netherlands\_quality\_in\_diversity\_strategy.pdf*
- **Paramètres passés en argument :**
  - Nombre de thèmes : 10
  - Nombre d'itérations : 500
  - Probabilité minimale : 0.5 (probabilité à partir de laquelle des thèmes seront supprimés)
  - $\alpha = 1.5$ , pour rappel une valeur élevée de  $\alpha$  (en l'occurrence  $> 1$ ) signifie qu'on suppose que les documents sont un mélange de différents thèmes.

La librairie *pyLDAvis* nous a permis d'afficher les résultats de manières assez explicites (figure 12 et figure 13), en l'occurrence elle nous offre une visualisation interactive des thèmes, leurs relations et leurs importances.

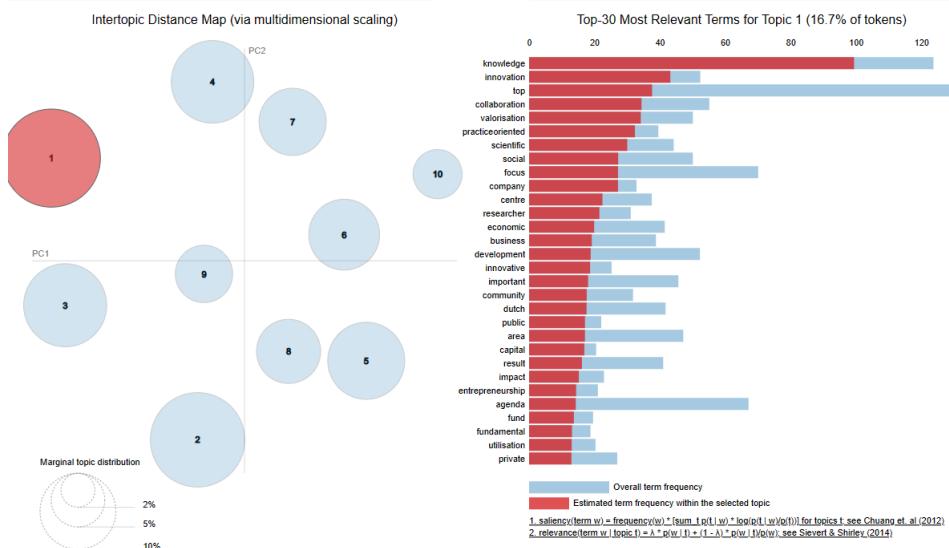


FIGURE 12 – Visualisation des thèmes extraits par la LDA : thème 1

Chaque bulle dans la partie des gauches des images va représenter un thème (représenté par une collection de mots) que le modèle a pu extraire, le rayon de la bulle va représenter à quel point un thème est pertinent dans le texte et enfin la distance entre deux bulles représenter à quel point les thèmes sont similaires. La partie de droite représente les 30 mots les plus pertinents par thème, ce qui nous donne une première idée du thème en les analysant.

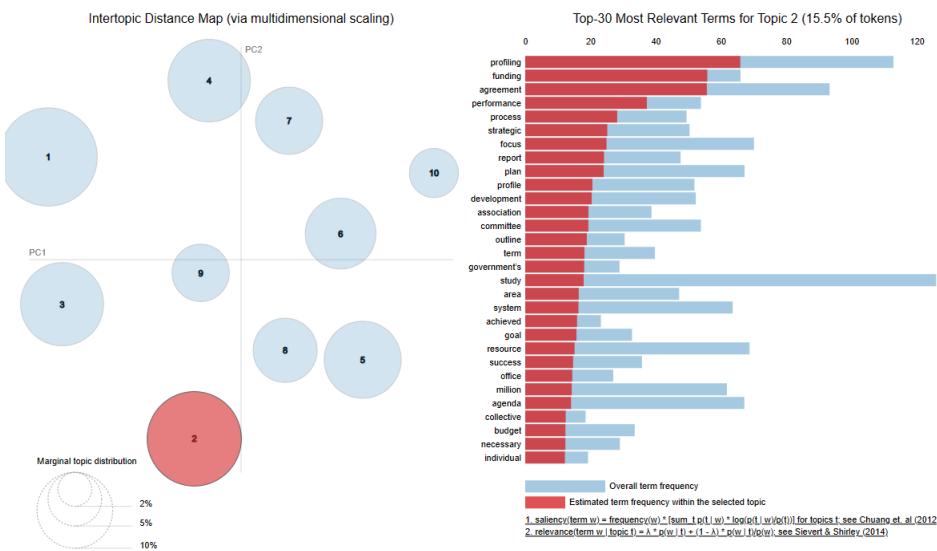


FIGURE 13 – Visualisation des thèmes extraits par la LDA : thème 2

**Premières observations :** En observant les figures, deux gros thèmes disjoints ont été identifiés, et en regardant les mots les plus pertinents de chaque thème, on peut remarquer que le thème 1 semble traiter de termes scientifiques (*knowledge, innovation, scientific, researcher, economic...*) tandis que le thème 2 semble traiter de termes financiers (*funding, agreement, plan, resource...*). Par conséquent, nous pouvons supposer que deux des thèmes abordés dans le document sont **STEM education** et **fundings and resources**.

Maintenant que nous avons un moyen d'extraire un ensemble de mots qui vont représenter un certain thème, on se pose la question de comment utiliser cette information pour avoir une représentation plus concrète des thèmes, pour ce faire on va utiliser des expressions régulières (REGEX).

### 6.3.2 • REGEX

La REGEX (ou expressions régulières) est un outil permettant de manipuler et d'extraire des données textuelles, elle permet de rechercher et de faire correspondre des motifs spécifiques avec une chaîne de caractère. D'un point de vue général, les expressions régulières peuvent être utilisées pour différentes tâches comme le nettoyage de données ou l'extraction d'informations.

Dans le contexte de *Topic Modeling* utilisant la *LDA*, on a utilisé la REGEX pour raffiner l'identification de thèmes pertinents dans le corpus de textes en utilisant un dictionnaire thèmes-mots que nous avons faits nous-mêmes et en faisant une recherche des mots pertinents extraits par la *LDA* dans ce dictionnaire.

Voici à quoi ressemble les premières lignes du dictionnaire :

```
themes = {
    "STEM_education": [ "STEM_education", "science", "technology", "mathematics" ... ],
    "early_childhood_education": [ "early_childhood_education", "preschool" ... ],
    "curriculum_and_instruction": [ "curriculum", "instruction", "teaching" ... ],
    "funding_and_resources": [ "funding", "resources", "budget", "financial" ... ]
}
```

Et donc pour chaque texte, on applique la *LDA* pour extraire un nombre prédéfini de thèmes (qui seront représentés par un ensemble de mots pertinents), on itère sur ces mots pertinents et à chaque fois qu'un des

mots apparaît, on incrémente le nombre d'apparitions de son thème correspondant. Les thèmes avec le plus d'apparitions seront les thèmes principaux du texte.

En appliquant ça sur le résultat de la LDA de l'exemple précédent, on obtient que les thèmes du texte sont **STEM education** et **funding and resources**, ce qui confirme bien la première analyse qu'on avait sur les résultats et qui nous permet donc d'avancer en combinant ces deux méthodes sur tout le jeu de données.

### 6.3.3 • APPLICATION À PLUS GRANDE ÉCHELLE

Maintenant qu'on a un modèle qui fonctionne dans le sens où il nous fournit un résultat sur les thèmes dominants dans un texte de l'éducation nationale, on peut itérer sur tous les textes de chaque continent et pouvoir faire une analyse beaucoup plus globale quant aux thèmes qui sont prédominants d'un point de vue continental.

On obtient l'histogramme de fréquence thématique suivant :

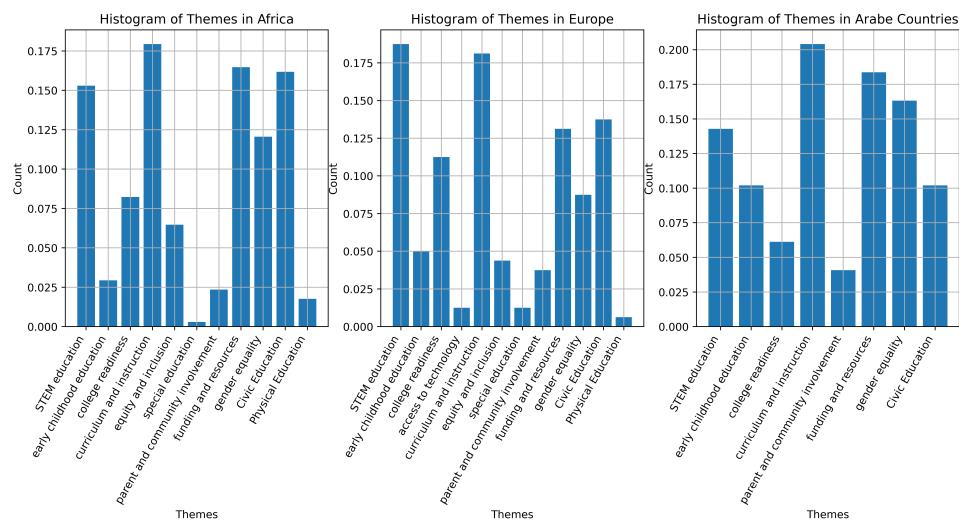


FIGURE 14 – Histogramme des fréquences d'apparition des thèmes en Afrique, Europe et Pays Arabes

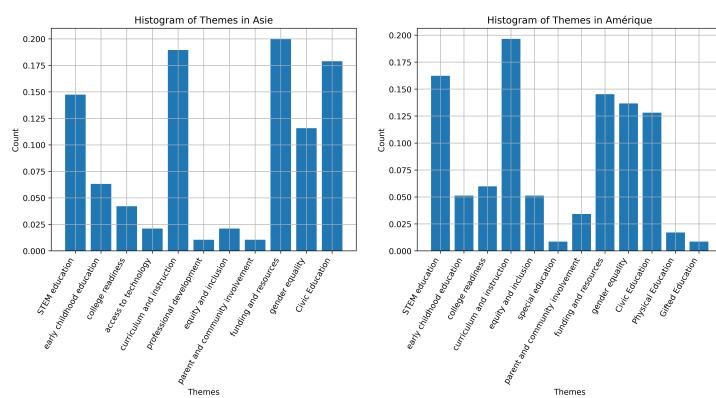


FIGURE 15 – Histogramme des fréquences d'apparition des thèmes en Asie et en Amérique

On peut tirer plusieurs conclusions quant aux résultats fournis par la figure 14 et 15. Premièrement, les thèmes **STEM education** et **curriculum and instruction** font partie des thèmes les plus communément

abordés dans les cinq continents, ce qui pourrait suggérer que ce sont des domaines qui sont considérés importants dans le développement éducatif global.

Deuxièmement, on retrouve plusieurs différences intercontinentales en termes de fréquences de certains thèmes, on peut par exemple remarquer qu'en Afrique les thèmes de financements et d'éducation civique sont beaucoup plus abordés que l'égalité des genres. En revanche, l'égalité des genres est un thème beaucoup plus commun dans les pays arabes, ce qui suggère que différentes régions priorisent différents domaines de l'éducation en fonction de leurs demandes spécifiques et des défis auxquelles elles ont pu faire face.

Globalement, ces résultats peuvent être utiles pour les décideurs politiques et éducateurs cherchant à développer et à améliorer l'éducation nationale. En comprenant les thèmes communs et les différences entre les différentes régions, ils peuvent mieux adapter leurs stratégies pour adresser les besoins propres à leurs populations.

## 7

## NAMED ENTITY RECOGNITION (NER)

NER (*Named Entity Recognition*) signifie Reconnaissance des Entités Nommées consiste à identifier et à classer les entités nommées dans un texte. Les entités nommées sont des mots ou des phrases spécifiques qui font référence à des objets du monde réel tels que des personnes, des lieux, des organisations, des produits, etc.

L'objectif de la NER est d'identifier et de classifier automatiquement ces entités nommées dans un texte, et de les assigner à des catégories prédéfinies telles que personne, organisation, lieu, date, heure, etc. Cela est utile pour nous pour capturer les mots les plus pertinents dans notre corpus de texte.

Une fois qu'un modèle est entraîné, il peut être utilisé pour identifier et classifier automatiquement les entités nommées dans un nouveau texte. Cela implique d'analyser le texte et d'identifier des séquences de mots qui correspondent aux modèles et aux caractéristiques appris pendant l'entraînement. Le modèle attribue ensuite les entités identifiées à leurs catégories respectives.

1. Extraire les entités "nommées".
2. Raccorder les entités pour former un nouveau corpus représentatif et nettement plus petit.
3. Exécuter la LDA sur le nouveau jeu de données.

Ousted **WeWork** founder **Adam Neumann** lists his **Manhattan penthouse** for **\$37.5 million**

[organization]

[person]

[location]

[monetary value]

FIGURE 16 – Exemple d'extraction d'entités.

Nous avons utilisé un modèle de *transformers* de NER pré-entraîné avec BERT de la librairie spaCy qui comporte plusieurs catégories significatives telles que les organisations, les lieux, l'argent, la date et les personnes. Nous avons appliqué cette méthode en remplaçant tous les mots d'un document par ceux qui appartiennent à la catégorie organisation (org), qui s'est avérée après plusieurs essais la plus pertinente. Bien que cette méthode ait réduit le nombre de mots dans le document, elle ne nous permet plus d'appliquer la méthode de comptage des mots du dictionnaire utilisée précédemment, car un dictionnaire adapté est nécessaire et les organisations peuvent varier d'un pays à l'autre.

La NER permet de réduire considérablement la taille du corpus traité tout en conservant relativement le sens. On peut voir la NER comme étant un *preprocessing* des données qui, couplé avec à la LDA, nous donne une démarche pour réussir une extraction beaucoup plus rapide des thèmes (Fig-17).

Cette méthode nous a fourni un ensemble de mots pour chaque thème qui sont plus pertinents pour le sujet, comme illustré ci-dessous. Cette approche nous a permis de mettre en évidence des priorités très spécifiques à un pays ou à une région qui n'ont pas pu être détectées avec le dictionnaire précédemment défini.

Le résultat donné sur l'ensemble des documents de l'Afrique :

- **Education system and its components** : "education", "school", "university", "health", "community", "public"
- **National education policies** : "national", "ministry", "institute", "children", "act", "government"

- **Development and planning** : "development", "department", "science", "planning", "technical", "technology"
- **International education** : "nations", "international", "council", "teachers", "research", "association"
- **Management and resources** : "management", "state", "examinations", "professional", "sciences", "fund", "economic", "world"

Le résultat donné sur l'ensemble des documents de l'Europe :

- **Education and Social Sciences** : "education," "social," and "science" are highly probable.
- **Organizational Structure** : "bureau," "department," "unioneuropean," and "parliament."
- **Research and Academia** : "ministry," "university," "research," and "government."
- **National Development** : "council," "development," "national," and "international."
- **European Union and Technology** : "european," "union," "technology," and "eu."

On peut donc voir qu'on obtient des résultats beaucoup plus raffinés en comparaison avec la LDA, les mots représentant les termes sont beaucoup plus précis. En plus d'une précision accrue, la NER permet de réduire radicalement le temps d'exécution de la LDA comme le montre le graphique ci-dessous.

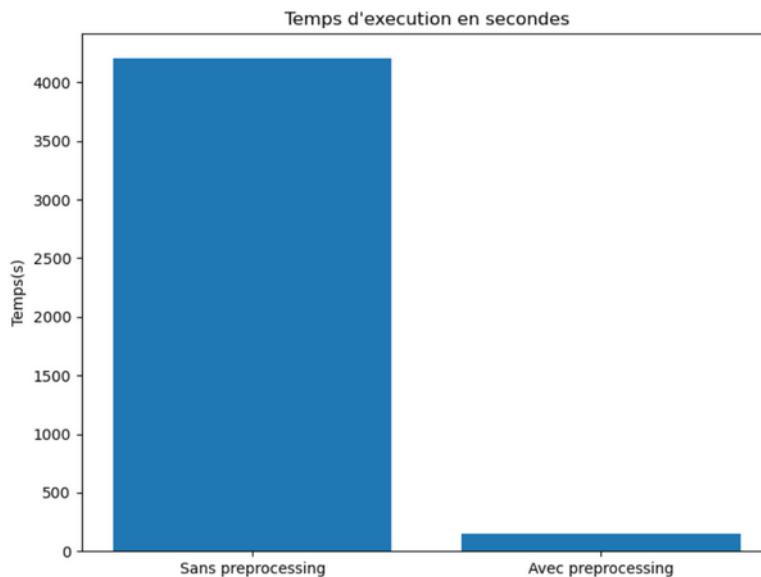


FIGURE 17 – Temps d'exécution en secondes de l'algorithme de LDA sans et avec le preprocessing NER

## 8

## CLASSIFICATION AVEC RÉSEAUX DE NEURONES RÉCURRENTS (RNN)

### 8.1 INTRODUCTION

Nous nous intéressons désormais à un autre pan du *Natural Language Processing* : les réseaux de neurones.

Les réseaux de neurones ont été inspirés du fonctionnement du cerveau humain et permettent à des machines d'exécuter différentes tâches telles que la compréhension de texte. L'avantage d'un réseau de neurones est qu'à l'inverse de la LDA/NER, le modèle peut être entraîné une bonne fois pour toutes, et qu'on peut accéder à nos résultats en temps constant après entraînement. Cependant, le désavantage est que c'est un modèle d'apprentissage supervisé, c'est-à-dire qu'il requiert des données labellisées. Dans notre cas, pour faire de la classification de textes, l'idéal serait d'avoir un *dataset* qui contient des phrases labellisées 1 si la phrase traite du thème en question, et 0 sinon.

### 8.2 RÉSEAUX DE NEURONES

#### 8.2.1 • INTRODUCTION

Un réseau de neurones est composé de couches de noeuds interconnectés. Chaque neurone reçoit des informations internes (depuis d'autres neurones) et externes (*input*). En ajustant les poids que l'on attribue à chacun des neurones, nous pouvons traiter des tâches aussi complexes que la classification de textes par thème.

Les couches sont classées en plusieurs parties :

- **La couche d'entrée** (*input*) : chacun des neurones de cette couche reçoit une variable d'entrée.
- **Les couches cachées** (*hidden layer*) : ces couches intermédiaires contiennent des neurones qui permettent de traiter et de transformer les données d'entrée en effectuant des transformations mathématiques.
- **La couche de sortie** (*output*) : C'est elle qui produit les prédictions ou classifications du problème considéré.

Les poids, qui sont des valeurs numériques modifiées au cours du processus d'apprentissage, relient les neurones d'un réseau. Chaque neurone reçoit un mélange linéaire pondéré de ses entrées à laquelle on ajoute un terme de biais. La sortie du neurone est ensuite créée en faisant passer cette somme pondérée par une fonction d'activation, telle que la fonction sigmoïde, la tangente hyperbolique ou la fonction ReLU (*Rectified Linear Unit*).

**Propriété 8.2.1.1** (Sortie d'un neurone). *On note  $N_m$  l'ensemble des neurones de la couche  $m$ . Soit  $m \in \mathbb{N}$  et  $j \in N_m$ . Notons  $s_{j,m}$  la sortie du neurone  $j$  dans la couche  $m$ . On note  $w_{ij}^m$  le poids que l'on associe entre le neurone  $i$  de la couche  $m - 1$  et le neurone  $j$  de la couche  $m$ . Soit  $b_j$  le biais associé au neurone  $j$  et  $f$  la fonction d'activation. Alors :*

$$s_{j,m} = f \left( \sum_{i \in N_{m-1}} (w_{ij}^m * y_{i,m-1}) + b_j \right) \quad (6)$$

### 8.2.2 • RÉTRO-PROPAGATION

L'objectif d'un réseau de neurones est de s'approcher le plus possible d'un prédicteur précis. Pour cela, il actualise chacun de ses poids par **rétro-propagation** en minimisant une fonction de perte qui mesure les écarts entre les valeurs prédites et les valeurs réelles. L'algorithme de rétro-propagation se base sur une méthode de descente de gradient expliquée précédemment, et emploie la *chain rule* des dérivées partielles pour actualiser les poids.

Voici les différentes étapes de la rétro-propagation :

— **Forward propagation :**

L'algorithme commence par déterminer la valeur en sortie du réseau de neurones (valeur prédite), puis la compare avec la valeur attendue (valeur théorique) grâce à une fonction d'erreur. Le réseau de neurones utilise alors les poids, biais et fonction d'activation qui lui sont attribués.

— **Rétro-propagation ou backpropagation :**

L'algorithme calcule le gradient de l'erreur en sortie par rapport aux différents poids et biais. Il remonte alors vers la couche d'entrée en calculant les différents gradients intervenant dans le calcul par dérivés composées (cf. exemple ci-dessous).

— **Actualisation des poids et des biais :**

L'algorithme actualise ses différents paramètres proportionnellement aux gradients calculés (descente de gradient à pas constant). Le pas de cette descente correspond au pas d'apprentissage : "*learning rate*".

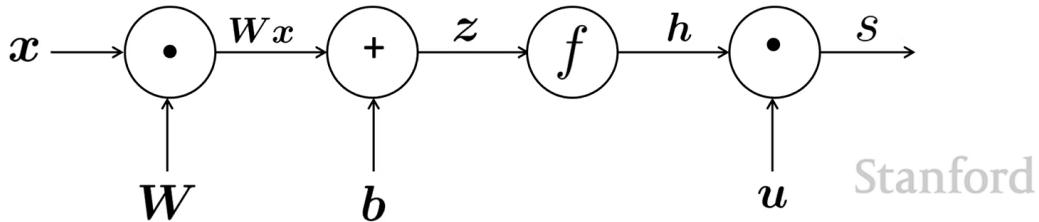


FIGURE 18 – Schéma d'un réseau composé d'une seule couche de neurones décomposée selon ses opérations.  
Source : Chris Manning, Stanford Winter 2023, CS224n

**Exemple 8.2.2.1** (Rétro-propagation). *Regardons la rétro-propagation à travers un exemple simple d'un réseau de neurones décomposé par ses opérations.*

*On a ici :*

- $x \in \mathbb{R}^d$  en entrée.
- $W \in \mathbb{R}^{d \times m}$  une matrice de poids.
- $b \in \mathbb{R}^m$  un terme de biais de sorte que :  $z = Wx + b$
- $f$  une fonction d'activation. On pose alors  $h = f(z) = f(Wx + b)$
- $s$  la sortie définie par :  $s = u^T \cdot h$
- $L$  la fonction de perte (de coût).

*La première étape est de calculer  $s$  grâce à l'entrée  $x$ . Ensuite, nous actualisons chacun des paramètres du réseau grâce au calcul de différents gradients.*

*Au temps  $n$  Pour chaque paramètre  $p_i^n$  du réseau, on actualise  $p_i^n$  telle que :  $p_i^{n+1} = p_i^n - \eta \frac{\delta L}{\delta p_i^n}$ .*

*Pour calculer  $\frac{\delta L}{\delta p_i^n}$ , on utilise la chain rule pour remonter dans le réseau. Par exemple :*

$$\frac{\delta L}{\delta b} = \frac{\delta L}{\delta s} \frac{\delta s}{\delta h} \frac{\delta h}{\delta z} \frac{\delta z}{\delta b}$$

Pour généraliser à tout un réseau, on applique cette méthode à toutes les couches à chaque entrée de l'étape d'entraînement, ce qui nous permet de minimiser globalement notre fonction de coût  $C$ .

## 8.3 RÉSEAUX DE NEURONES RÉCURRENTS

### 8.3.1 • PRÉSENTATION

Les réseaux de neurones récurrents sont une branche des réseaux de neurones spécialisée pour traiter des données séquentielles ou temporelles. En effet, ces réseaux offrent une sorte de "mémoire" qui est utile pour appréhender des problématiques de séries temporelles, ou de *NLP*.

Dans un réseau de neurones récurrent, les neurones du *hidden layer* sont non seulement reliés aux neurones des couches précédentes et suivantes, mais aussi à eux-mêmes par des liaisons *récurentes*. Cela permet de conserver une partie de l'information transmise pour, par exemple, comprendre un contexte d'un mot. Ainsi, ces neurones s'avèrent particulièrement intéressantes pour traiter de classification de textes.

**Définition 8.3.1.1** (État caché d'un neurone dans un RNN). *Dans un réseau de neurones récurrent, nous pouvons introduire la notion d'état d'une couche puisque celle-ci "évolue dans le temps". Pour une couche donnée  $h$ , nous noterons  $h_t$  son état à l'instant  $t$ .*

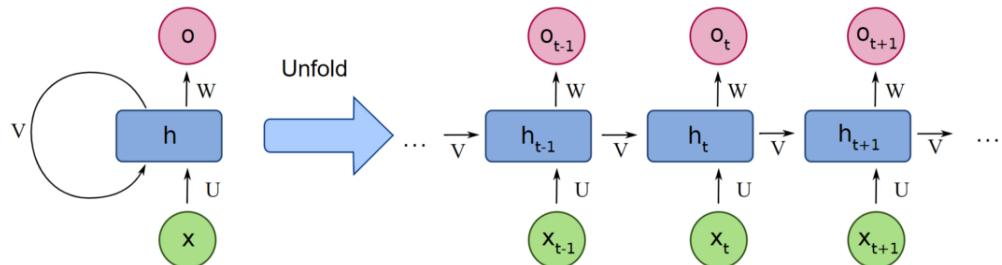


FIGURE 19 – Schéma simple d'une couche d'un réseau de neurones récurrents. Source : medium.com

### 8.3.2 • PROBLÈME DE LA DISPARITION DU GRADIENT

Néanmoins, dans certains cas, l'entraînement du réseau de neurones est délicat. Il existe parfois un problème inhérent à la structure de neurones et de l'entraînement par rétro-propagation : le problème de la disparition ou de l'explosion du gradient.

En effet, lors de répétitions successives de rétro-propagation pour actualiser les poids de nos neurones, nous pouvons nous heurter à des gradients de plus en plus petits (ou à l'inverse, grand), ce qui n'aura quasi plus d'effet sur l'actualisation de nos poids. Pour des réseaux de neurones profonds en particulier, le problème est flagrant : pour calculer les dérivées partielles de nos paramètres, nous utilisons la formule de composition de dérivées partielles. Après un temps long, pour actualiser l'état  $h_t$  de la couche (ou du neurone), le produit dans la formule de dérivées partielles devient de plus en plus longue. Or, si chaque gradient qui intervient dans le produit est en norme inférieure à 1, le gradient final sera très petit.

**Propriété 8.3.2.1** (Disparition du gradient). *On note :*

- $h_1, h_2, h_3 \dots$  des états successifs d'une couche  $h$ .
- $x_1, x_2, x_3 \dots$  une série d'inputs de la couche.
- $s_1, s_2, s_3 \dots$  une série d'outputs de la couche.
- $L$  la fonction de perte (coût), paramétrée par  $\theta$ .

On a alors :

$$(s_t, h_t) = F(h_{t-1}, x_t, \theta)$$

On fait l'hypothèse simplificatrice que la couche est représentée par sa sortie :  $h_t = s_t$ .

D'où :  $s_t = F(s_{t-1}, x_t, \theta)$

La formule d'actualisation de  $\theta$  est donnée par :

$$\Delta\theta = -\eta \cdot [\nabla_s L(s_t) \frac{ds_t}{d\theta}] \quad (7)$$

Or

$$\frac{ds_t}{d\theta} = \nabla_s F(s_{t-1}, x_t, \theta) + \nabla_s F(s_{t-1}, x_t, \theta) \nabla_\theta F(s_{t-2}, x_{t-1}, \theta) + \dots$$

Donc :

$$\Delta\theta = -\eta \cdot [\nabla_s L(s_t) (\nabla_s F(s_{t-1}, x_t, \theta) + \nabla_s F(s_{t-1}, x_t, \theta) \nabla_\theta F(s_{t-2}, x_{t-1}, \theta) + \dots)] \quad (8)$$

La disparition du gradient provient de :  $\nabla_s F(s_{t-1}, x_t, \theta) \nabla_s F(s_{t-2}, x_{t-1}, \theta) \nabla_s F(s_{t-3}, x_{t-2}, \theta) \dots$

### 8.3.3 • LSTMs

Ainsi, Hochreiter et Schmidhuber en 1997 ont inventé une structure de neurones particulière qui parvient à passer outre le problème de la disparition du gradient : les LSTM (*Long Short-Term Memory*). Chaque LSTM est en fait divisée en quatre couches de neurones. L'idée principale est que la cellule LSTM rajoute une variable d'état qui permet de renforcer sa mémoire sur le long terme.

**Définition 8.3.3.1** (Cellule d'état (*state cell*)). On note  $C_t$  la state cell qui indique la "mémoire" de la cellule au temps  $t$ .

Voici le schéma complet d'une LSTM :

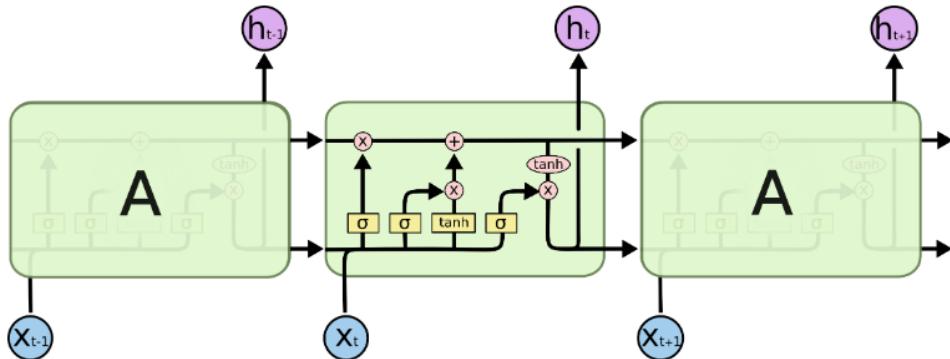


FIGURE 20 – Schéma d'une cellule LSTM au cours du temps. Source : colah.github.io

En jaune, nous apercevons les quatre couches de neurones de la cellule LSTM. Nous allons maintenant parcourir toute la cellule pour l'expliquer (se référer à l'annexe pour les schémas).

— La **porte d'oubli ou forget gate** :

La "porte d'oubli" décide quelles informations il faut garder dans l'entrée  $x_t$  et dans l'état caché précédent  $h_{t-1}$ . Le tout est passé dans une opération sigmoïde  $\sigma$  qui pondère entre 0 et 1 les entrées.

— La **porte d'entrée ou input gate** :

Elle décide des informations que l'on veut actualiser. On pondère  $x_t$  par une sigmoïde et on rajoute des nouveaux candidats potentiels grâce à une fonction tanh.

— La **porte de sortie ou output gate** :

C'est la porte qui décide quelle partie de l'information contenue dans  $x_t$  et dans  $C_t$  doit être transmise à l'état caché suivant et à la couche suivante.

## 8.4 APPLICATION AU CORPUS

---

### 8.4.1 • CRÉATION DU DATASET

Comme les RNN sont une méthode d'apprentissage supervisée, nous devons nous-mêmes créer un set de données sur lequel entraîner notre modèle. Nous avons donc eu l'idée de labelliser un ensemble de phrases grâce à un algorithme de reconnaissance de mots. Plus précisément, pour chaque thème en rapport avec l'éducation, nous avons créé un dictionnaire de mots (comme celui employé dans la partie précédente), et nous posons le fait que si une phrase traite du thème de l'éducation, celle-ci contiendra nécessairement un mot en rapport avec l'éducation. Ainsi, nous avons pu obtenir un ensemble de données telles que :

"Education is key to success" serait labellisée 1.

"I ate pasta this evening" serait labellisée 0.

Il existe beaucoup de problèmes liés avec cette façon de faire : en effet, le dictionnaire que nous écrivons n'est clairement pas exhaustif, et nous pouvons passer à côté de mots qui ont un rapport avec l'éducation. Deuxièmement, nous pouvons très bien parler des cours sans évoquer des mots qui ont un sens en rapport avec l'éducation. Par exemple : "j'ai bien réussi" pourrait aussi bien parler d'un examen, que d'un plat. Enfin, même après avoir constitué notre set de données, l'ensemble n'est pas équilibré. C'est un gros problème pour notre modèle d'entraînement et cela va se refléter dans les résultats.

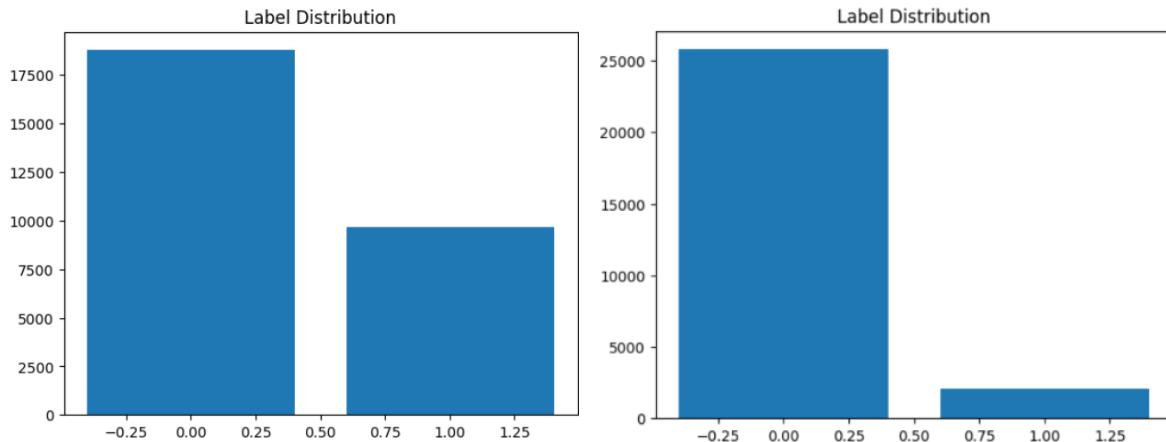


FIGURE 21 – À gauche, la distribution des labels pour le thème "education", à droite, pour le thème "STEM education"

### 8.4.2 • RÉSULTATS DU MODÈLE

En ce qui concerne le modèle, nous en avons essayé plusieurs, et celui qui fonctionne le mieux en termes de précision est le suivant : une couche de LSTMs, puis deux couches de neurones "classiques".

Voici les différents graphiques d'apprentissage des modèles :

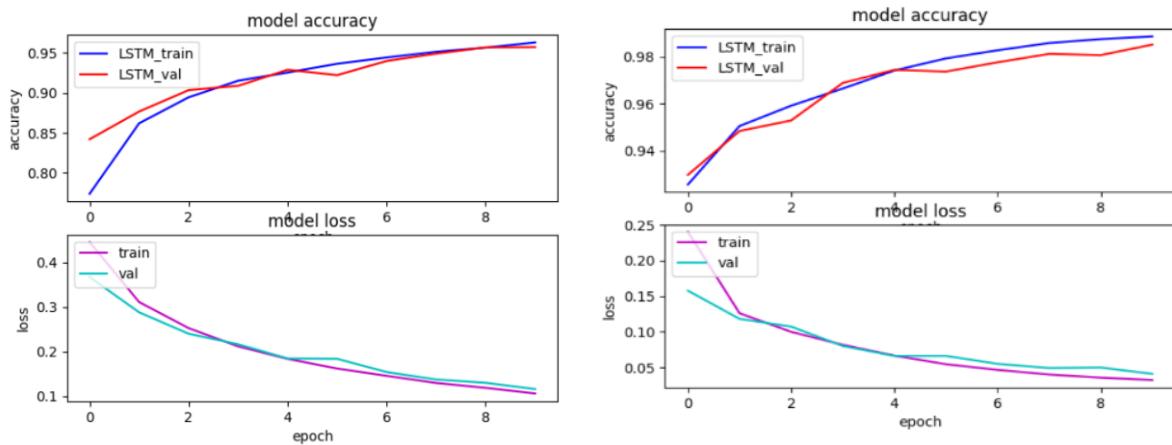


FIGURE 22 – À gauche, la courbe d'apprentissage pour le thème "education", à droite, pour le thème "STEM education"

En ce qui concerne le *dataset*, on observe qu'il n'est pas du tout équilibré : cela va d'un facteur deux à un facteur dix en termes de labels négatifs en plus. Cela va poser un problème dans nos matrices de confusion (Fig. 23).

En ce qui concerne l'apprentissage du modèle, on voit que notre fonction de perte se minimise assez bien, et que notre précision ne fait qu'augmenter au fur et à mesure que notre modèle apprend.

Malgré une précision très élevée, on peut remarquer un problème assez inquiétant sur les matrices de confusion :

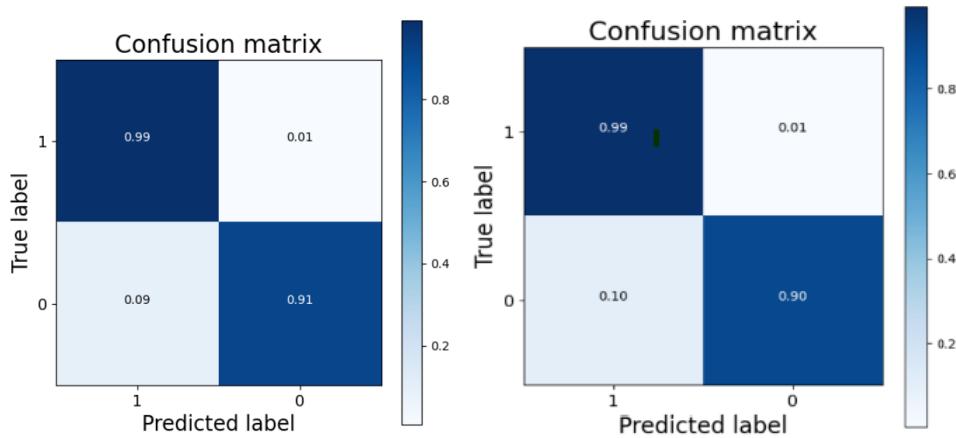


FIGURE 23 – À gauche, la matrice de confusion pour le thème "education", à droite, pour le thème "STEM education"

On observe que l'erreur faux-positif est de l'ordre de 10 fois plus grande que l'erreur faux-négatif, ce qui n'est clairement pas satisfaisant. Les erreurs de notre modèle proviennent en grande partie de notre dataset qui n'est pas vraiment adapté à notre problème. Cependant, avec les autres modèles que l'on a implémentés, une reconstruction du dataset peut être envisagé, auquel cas le modèle RNN peut tout à fait s'avérer pertinent.

## 9

# CONCLUSION

---

Nous avons donc réussi à implémenter différentes méthodes qui permettent l'extraction de thèmes dans les textes

- Doc2Vec : bien que facile à implémenter et à interpréter, cette méthode reste très naïve et imprécise.
- GloVe & *clustering* : ce procédé donne des résultats encore imprécis, même s'ils sont exploitables. On peut effectivement trouver l'apparition de certains thèmes dans les différents textes. Cependant, l'humain doit vraiment être attentif aux différents mots proposés pour représenter les *clusters*. Le procédé demande aussi une bonne puissance de calcul.
- Topic Modelling, LDA et NER : c'est la méthode la plus efficace qui émerge de ce rapport. Nous avons fourni des résultats tout à fait exploitables, et un algorithme qui ne nécessite pas un temps d'exécution très élevée.
- RNN et LSTM : procédé encore imprécis notamment à cause du manque de données d'entraînement. Une façon d'y remédier serait d'utiliser les techniques précédemment évoquées pour créer un jeu de données d'entraînement que l'on pourrait utiliser pour entraîner notre réseau de neurones.

Globalement, ce sujet fut très intéressant, car il nous a permis d'étudier une vaste facette du *NLP*. Il nous a sensibilisé aux problématiques actuelles qu'un chercheur en *deep learning* connaît telles que les questions du traitement et du manque de données.

Après une présentation de nos travaux auprès de l'UNESCO-IIPE quelques améliorations, notamment dans l'utilisation courante de nos algorithmes, se dégagent :

- La mise en place d'une interface *user-friendly*. Pour faciliter l'exploitation des méthodes précédentes, une application web a été rapidement développée et permet de drag&drop des pdf pour leur analyse au moyen de la représentation Doc2Vec.
- Suite à une discussion avec les chercheurs de l'UNESCO-IIPE, une exploitation des métadonnées des pdf tel que les dates de parution des textes, serait souhaitable pour avoir une classification plus raffinée et observer l'évolution des tendances au fil des années.
- Enfin, améliorer les différentes étapes de tout le processus, traitement comme analyse. Pour le prétraitement, l'exploitation des techniques de VrDU serait souhaitable pour, à la fois mieux "lire" le document, mais également récupérer des informations supplémentaires sur sa structure pour complexifier l'analyse. Au sujet de l'exploitation, obtenir un dataset d'entraînement permettrait de mieux qualifier les performances de nos modèles et d'entrainer des réseaux de neurones. Obtenir un dictionnaire de mots et de thèmes précis renforcerait également les phases de pré-traitement et d'analyse.

## RÉFÉRENCES

---

- [1] V Kishore Ayyadevara. 2018. Recurrent Neural Network. In : *Pro Machine Learning Algorithms : A Hands-On Approach to Implementing Algorithms in Python and R*. Apress, Berkeley, CA. pp. 217-257.
- [2] Bird, Steven and Klein, Ewan and Loper, Edward. 2009. *Natural Language Processing with Python*. O'Reilly Media, Inc.
- [3] David M. Blei, Andrew Y. Ng and Michael I. Jordan. 2003. In : *Latent Dirichlet Allocation*.
- [4] Alberto Bietti. 2012. In : *Latent Dirichlet Allocation*.
- [5] Radim Řehůřek, Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In : *Proceedings of the LREC 2010 Workshop on New Challenges for NLP*. Malta.
- [6] Jeffrey Pennington and Richard Socher and Christopher D. Manning. 2014. GloVe : Global Vectors for Word Representation. In : *Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar. pp. 1532-1543.
- [7] Jin, X., Han, J. (2011). K-Means clustering. In : Sammut, C., Webb, G.I. (eds) *Encyclopedia of Machine Learning*. Springer, Boston
- [8] Steve Oudot. Spring 2023. clustering with k-Means. In : *Algorithmes pour l'analyse de données en C++*. Ecole polytechnique. Palaiseau, France.
- [9] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In : *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*. Institute for Computer Science, University of Munich. pp 226–231.
- [10] JiayuLin. 2016. *On The Dirichlet Distribution*. Queen's University Kingston, Ontario, Canada.
- [11] Arun Pandian. 2020. *NLP Beginner - Text classification using LSTM*. Kaggle.
- [12] Grégoire Allaire, Alexandre Ern. Spring 2023. Algorithmes d'optimisation. In : *Optimisation et contrôle*. Ecole polytechnique. Palaiseau, France. pp. 59-63
- [13] Arya Roy. 2021. *Recent Trends in Named Entity Recognition (NER)*. Cornell University.
- [14] Chris Manning. Winter 2023. Natural Language Processing with Deep Learning. Stanford, California, United-States.
- [15] Ralf C. Staudemeyer, Eric Rothstein Morris. 2019. *Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks*. Cornell University

## 10

# ANNEXES

---

### ANNEXE - BIBLIOTHÈQUES UTILISÉES

---

- PDF plumber : Singer-Vine, J., & The pdfplumber contributors. (2023). pdfplumber (Version 0.9.0) [Computer software]. <https://github.com/jsvine/pdfplumber>
- Gensim : Rehurek, R., & Sojka, P. (2011). Gensim—python framework for vector space modelling. NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic.
- Sklearn : Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesnay. 2011. Scikit-learn : Machine Learning in Python. In : *Pedregosa et al., JMLR 12*, pp. 2825-2830.
- Numpy : Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. In : *Nature* 585. pp 357–362.
- Pandas : McKinney, W., & others. (2010). Data structures for statistical computing in python. In : *Proceedings of the 9th Python in Science Conference* (Vol. 445, pp. 51–56).
- SpaCy : Honnibal, M., & Montani, I. (2017). *spaCy 2 : Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing*.
- Tensorflow : Abadi, Mart&x27;in, Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... others. (2016). Tensorflow : A system for large-scale machine learning. In : *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. pp. 265–283
- Keras : Chollet, F., & others. (2015). Keras. GitHub. Retrieved from <https://github.com/fchollet/keras>
- nltk : Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python : analyzing text with the natural language toolkit*. Reilly Media, Inc.

## ANNEXE - FONCTIONNEMENT D'UNE LSTM

Toutes les figures utilisées viennent de colah.github.io.

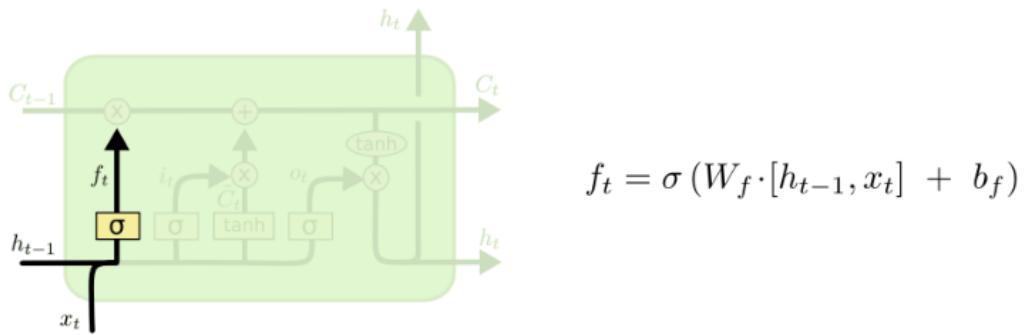


FIGURE 24 – Forget gate

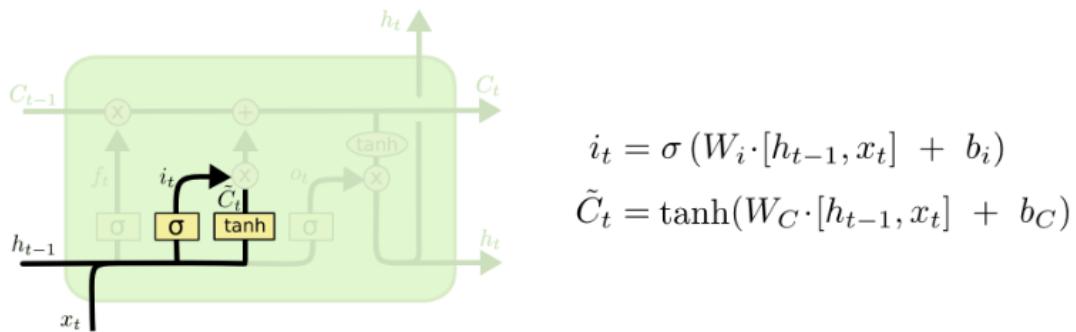


FIGURE 25 – Input gate

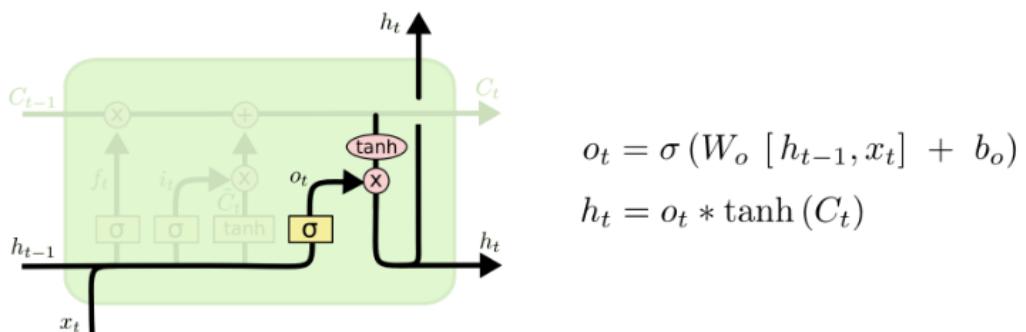


FIGURE 26 – Output gate