RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de Ghardaïa

Faculté des Sciences et de la Technologie

Département des Mathématiques et d'Informatique

Laboratoire de Mathématiques et Sciences Appliquées

MÉMOIRE

Présenté pour l'obtention du **diplôme** de **MASTER**

En **Informatique**

Spécialité : Systèmes Intelligents pour l'Extraction des Connaissances (SIEC)

Présenté par : Mohammed Mounsif BELLAOUAR & Issam Eddine GHADA

Thème

Modélisation thématique : cas des publications scientifiques

Membres de jury

M. Slimane OULADNAOUI	MCB	Univ. Ghardaia	Président
M. Abderahmane ADJILA	MAA	Univ. Ghardaia	Examinateur
M. Slimane BELLAOUAR	MCB	Univ. Ghardaia	Encadrant

Année Universitaire 2019/2020

ملخص

يتميز مجتمعنا للمعلومات بوفرة المعلومات الناتجة عن زيادة الرقمنة. بشكل عام، تكون هذه المعلومات في شكل نص غير موسوم ولا يمكن أن يُنسب دائما إلى مجال موضوعي معين. وهذا يجعل مهمة الحصول على رؤية موضوعية لمجموعة المعلومات تحديا صعبا. وبالتالي، قد يكون من المفيد اللجوء إلى خوارزميات غير خاضعة للإشراف للتعامل مع نمذجة الموضوع. نمذجة الموضوع هي مجال يهتم بتحليل النص لالتقاط معنى المصطلحات وفقا لسياقها في لغة طبيعية.

في مذكرتنا، نقدم مفهوم نمذجة الموضوع، والمناهج الملازمة لها وكذلك مجالات التطبيق. على مستوى العمل التجريبي، نجري دراسة مقارنة بين طريقة التحليل الدلالي الكامن LSA وتخصيص ديريتشيلت الكامن LDA باستخدام مجموعة المقالات العلمية من مؤتمر NIPS. النتائج المتحصل عليها من حيث وقت التنفيذ والاتساق الموضوعي كانت لصالح طريقة LDA. .

الكلمات المفتاحية : نمذجة الموضوع، LDA، LSA، المنشورات العلمية.

Résumé

Notre société d'information est caractérisée par une surabondance d'information résultant d'une digitalisation grandissante. Généralement, cette information est sous forme de texte non étiqueté que l'on ne peut pas toujours attribuer à un certain domaine thématique. Ceci rend la tâche d'avoir une vision thématique d'une collection d'informations un défi difficile.

Ainsi, il pourrait être utile de faire recours à des algorithmes non supervisés pour aborder la modélisation thématique.

Une telle modélisation s'intéresse à l'analyse de texte pour capturer le sens des termes en fonction de leurs contextes dans un langage naturel.

Dans notre mémoire, nous introduisons le concept de la modélisation thématique, ses approches inhérentes ainsi que ses domaines d'applications.

Au niveau du travail expérimental, nous conduisons une étude comparative entre la méthode d'analyse sémantique latente (Latent Semantic Analysis, LSA) et celle d'allocaton de Dirichlet latente (Latent Dirichlet Allocation, LDA) en utilisant le corpus des articles scientifique de la conférence NIPS.

Les résultats obtenus en terme de temps d'exécution et de cohérence thématique sont en faveur de la méthode LDA.

Mot clés : Modélisations thématiques, LSA, LDA, publications scientifiques

Abstract

Our information society is characterized by an overabundance of information which results from a growing digitization. Generally this information is in the form of unlabeled text which we cannot strictly attribute to a thematic domain. This makes the task of setting a thematic vision for a collection of data a difficult challenge.

Thus, it might be useful to resort to unsupervised algorithms to tackle the topic modeling problem.

Such a modeling is interested in analyzing text to capture the sens of the terms with respect to their contexts in the natural language.

In this thesis, we introduce the concepts of topic modeling, the inherent approaches as well as the application domains.

As far as the experimental work, we deploy a comparative study between the latent semantic analysis method (LSA) and the latent Dirichlet allocation method (LDA) using the corpus of the scientific articles of the NIPS conference.

In terms of running time and topic coherence, the obtained results are in favor of the LDA method.

Keywords : Topic modeling, LSA, LDA, Scientific publications

Dédicaces

Je dédie ce modeste travail à :

A mes très chers parents à qui je ne rendrais jamais assez pour leurs amours, leurs sacrifices, patiences et affections envers moi, leurs soutiens et soucis pour mon avenir et à qui je dois mes sincères et profonds remerciements.
A ma très chers sœurs et frères et Mon proche Personne pour leur appui et leur encouragement permanent, et leur soutien moral.
A tout ma famille et tous mes amis pour leur soutien tout au long de mon parcours universitaire.
A toutes mes amies, particulièrement : Abderazzak, Abderahime, Mohammed, Brahim et Issam.
Merci d'être toujours là pour moi.

Mounsif

Dédicaces

Avec une immense joie je dédie ce modeste travail :

A mes chers parents, pour tous leurs sacrifices, leur amour, leur tendresse, leur soutien et leurs prières tout au long de mes études,
A mes chères sœurs Kawthar et Tasnim pour leurs encouragements permanents, et leur soutien moral,
A mes chers frères Mouad et Sohaib pour leur appui et leur encouragement,
A toute ma famille pour leur soutien tout au long de mon parcours d'étude,
A toutes mes amies, particulièrement : Abderazzak, Abdesslam, Djelloul, Mounsif, Mohammed et Yacine.
Que ce travail soit l'accomplissement de vos vœux tant allégués, et le fuit de votre soutien infaillible,
Merci d'être toujours là pour moi

Issam

Remerciement

Nous tenons tout d'abord à remercier Dieu le tout puissant et miséricordieux, qui nous a donné la force et la patience d'accomplir ce modeste travail.

En second lieu, nous tenons à remercier notre encadreur le Dr Slimane BELLAOUAR, pour ses précieux conseils et aides durant toute la période de notre travail

Enfin, nous tenons également à remercier toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail.

Table des matières

Liste de tables

Table des figures

Listings

Introduction générale

Depuis la troisième génération des sites web, la priorité est donnée à la facilité de la production de l'information, sa personnalisation et son accessibilité. Ceci a donné naissance au phénomène de l'explosion de l'information caractérisant une société dite de l'information.

Les approches classiques d'analyse de données sont adaptées à des volumes de données restreints et marquent, ainsi, leurs limitation vis-à-vis de cette surabondance d'information. Les remèdes peuvent être multiples, une solution consiste à faire appel à des méthodes intelligentes d'analyse de données.

Dans notre contexte, où l'information traitée est de type texte non étiqueté, le défi est difficile. Il n'est pas toujours aisé d'attribuer cette information à un certain domaine thématique. Ainsi, il pourrait être utile de faire recours à des algorithmes non supervisés pour aborder la modélisation thématique.

La modélisation thématique est une classe des méthodes d'analyse de texte qui s'intéresse à l'analyse d'un sac de mots pour capturer le sens des termes en fonctions de leurs contextes dans un langage naturel.

Dans ce mémoire, nous introduisons le concept de la modélisation thématique, ainsi que les notions sous-jacentes. Par la suite, nous étudions les différents approches de la modélisation thématique dans la littérature. L'analyse sémantique latente (Latent Semantic Analysis, LSA) [Deerwester et al., 1990] représente le corpus sous forme d'une matrice terme-document puis procède à une réduction de la dimensionalité en utilisant la méthode de décomposition en valeurs singulières (Singular Value Decomposition, SVD).

La méthode d'analyse sémantique probabiliste (Probabilistic Latent Sementic Analysis, PLSA) [Hofmann, 1999] a pour objectif de garantir une base statistique solide à LSA.

La méthode d'allocation de Dirichlet latente (Latent Dirichlet Allocation, LDA) [Blei et al., 2003] est une méthode d'analyse sémantique probabiliste qui tente de découvrir les thématiques en considérant des connaissances supplémentaires préalables et en utilisant la distribution de Dirichlet.

La méthode lda2vec [Moody, 2016] est une méthode de modélisation thématique utilisant l'apprentissage profond. En effet, elle combine la technique de représentation du texte word2vec et la méthode LDA. Elle s'appuie spécifiquement sur le modèle skip-gram qui apprend l'incorporation d'un terme en essayant d'utiliser le terme d'entrée pour prédire les termes de contexte.

En tant que contribution, nous conduisons une étude empirique comparative entre la mé-

thode d'analyse sémantique latente (LSA) et celle d'allocation de Dirichlet latente (LDA) en utilisant le corpus des articles scientifique de la conférence NIPS (Neural Information Processing Systems). Les résultats obtenus en terme de temps d'exécution et de cohérence thématique sont en faveur de la méthode LDA.

Le présent mémoire est organisé comme suit :

Le Chapitre 1 se focalise d'abord sur la présentation des fondements théorique de la probabilité nécessaires pour la compréhension de la suite de ce mémoire. La deuxième partie de ce chapitre s'intéresse aux approches de présentation du texte.

Le Chapitre 2 se propose d'introduire le concept de la modélisation thématique. Ensuite, le chapitre procède à une étude bibliographique des différents approches de la modélisation thématique. Le chapitre se clôture par l'exploration des différents domaines d'application de la modélisation thématique, parmi lesquelles le domaine de la publication scientifique.

Le Chapitre 3 est réservé à notre contribution. Il sert à décrire étape par étape le déroulement de l'étude empirique comparative entre la méthode LSA et LDA en utilisant un corpus de publications scientifiques.

Enfin, la conclusion clôture notre mémoire en rappelant notre problématique et en dressant un bilan de notre travail ainsi que des perspectives à envisager.

Chapitre 1

Généralités

Dans ce chapitre nous présentons les informations nécessaires pour la compréhension de la suite de ce travail.

Nous nous focalisons d'abord sur les concepts de base de probabilité utilisées dans l'approche probabiliste de la modélisation thématique. Ensuite, nous traitons la représentation du texte, que se soit sous forme de *sac de mots* ou *word2vec* qui sont utilisés par les différentes approches de la modélisation thématique.

1.1 Probabilités : concepts de base

Dans cette section, nous introduisons juste les concepts de base utilisés dans la Section 2.2. Pour plus de détails sur la théories des probabilités avec des exemples, le lecteur peut consulter l'ouvrage de Rick Durrett [Durrett, 2019].

1.1.1 Modèles probabilistes

Un modèle probabiliste (ou un espace de probabilité) est une description mathématique d'une situation incertaine. Il consiste en une expérience, un espace des possibles et une loi de probabilité. Chaque modèle probabiliste implique un processus sous-jacent appelé expérience.

L'ensemble de tous les résultats possibles d'une expérience est appelé espace des possibles, ou encore univers, noté Ω. Un évènement est un sous-ensemble de l'univers Ω.

Une loi de probabilité attribue à chaque évènement A un nombre non négatif $P(A)$ appelé probabilité de l'évènement A. Une loi de probabilité doit vérifier les axiomes suivants :

— Non négativité : $P(A) \geq 0$ pour chaque évènement A.

— Additivité : si A et B sont deux évènement disjoints, alors

$$P(A \cup B) = P(A) + P(B)$$

Ceci peut être généralisé pour une séquence d'évènements disjoints.

$$P(A_1 \cup A_2 \cup ...) = P(A_1) + P(A_2) + ...$$

— Normalisation : $P(\Omega) = 1$.

1.1.2 Probabilité conditionnelle

La probabilité conditionnelle est un concept fondamental dans le cadre de la modélisation probabiliste. Elle permet de prendre en compte l'information dont on dispose (qu'un évènement B réalisé) pour actualiser la probabilité d'un évènement A.

Soient A et B deux évènements avec $P(B) \neq 0$, la probabilité conditionnelle de l'évènement A sachant l'évènement B notée $P(A|B)$ est définie par :

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

1.1.3 Formule de probabilité total (système complet d'évènements)

Une partition de Ω est un ensemble de parties de Ω deux à deux disjoints et dont la réunion est Ω.

Si $A_1, A_2, \ldots, A_n$ forme une partition de Ω alors pour tout évènement $B \subset \Omega$ on a :

$$P(B) = P(B \cap A_1) + P(B \cap A_2) + \ldots + P(B \cap A_n)$$

$$= P(A_1)P(B|A_1) + P(A_2)P(B|A_2) + \ldots + P(A_n)P(B|A_n)$$

En effet, il est parfois utile de décomposer un évènement en sous ensemble d'évènements élémentaires afin d'en calculer la probabilité d'une manière plus facile. C'est bien le principe de diviser et conquérir qui est très utile dans la modélisation et la résolution des problèmes.

1.1.4 Formule de Bayes

Soient $A_1, \ldots, A_n$ une partition de Ω et $B \subset \Omega$, tel que $P(B) > 0$. Alors pour tout $1 \leq i \leq n$,

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_{j=1}^{n} P(B|A_j)P(A_j)}$$

Il n'est pas difficile de vérifier la validité de la formule de Bayes en appliquant la formule de la probabilité conditionnelle et celle de la probabilité totale.

1.1.5 Indépendance

Deux évènements A et B sont indépendants si et seulement si $P(A \cap B) = P(A).P(B)$.

La propriété d'indépendance de deux évènements A et B se manifeste par la relation

$$P(A|B) = P(A)$$

ou

$$P(B|A) = P(B)$$

.

Aussi, l'indépendance peut se généraliser à n évènements :

$$P(\bigcap_{i=1}^{n} A_i) = \prod_{i=1}^{n} P(A_i) \tag{1.1}$$

1.1.6 Variables aléatoires

Une variable aléatoire est une application $X : \Omega \to E$

À chaque évènement particulier $A \in \Omega$, correspond une valeur $X(A) \in E$. Le plus souvent E est un partie de $\mathbb{R}$ ou de $\mathbb{R}^n$.

Les variables aléatoires peuvent être discrètes ou continues.

1.1.7 Distributions de probabilité

La distribution ou fonction de répartition ou loi d'une variable aléatoire X et la fonction $F_X(x) = P(X \leq x)$.

Cette fonction est définie par toutes les valeurs de x et prend des valeurs entre 0 et 1.

Elle est décroissante, elle tend vers 0 si $x \to -\infty$ et vers 1 si $x \to +\infty$

Distributions de Bernoulli

Une variable aléatoire discrète qui ne prend que les valeurs 0 et 1 avec des probabilités respectives p et $q = 1 - p$ est appelée variable de *Bernoulli*.

On utilise une variable de Bernoulli lorsqu'on effectue une épreuve qui n'a que deux issues (succès on échec). Une telle expérience est alors appelée épreuve de Bernoulli.

On affecte alors 1 à la variable en cas de succès et 0 en cas d'échec.

La distribution de probabilité est donnée par :

$$F_X(x) = P(X = x) = \begin{cases} p & \text{si } x = 1, \\ q = 1 - p & \text{si } x = 0. \end{cases}$$

On note cette distribution $X \sim \beta(p)$. Cette distribution ne dépend que d'un seul paramètre p, la probabilité de succès.

Distributions Binomiale

La loi *binomiale* permet de modéliser une épreuve de Bernoulli répétée n fois de façon indépendante.

On considère la variable aléatoire $X = X_1 + X_2 + \ldots + X_n$. Cette variable aléatoire désigne le nombre de succès lors de n épreuves. La variable X peut prendre alors $n+1$ valeurs : $0, 1, 2, \ldots, n$.

Cherchons $P(X = k)$, la probabilité d'obtenir k succès. La probabilité d'avoir k succès et $(n-k)$ échecs est toujours égale à $p^k q^{n-k}$.

Combien y en a-t-il d'évènements élémentaires qui composent l'évènement $(X = k)$? Il suffit de choisir k places de succès parmi les n possibles et les $n-k$ échecs prendront les places restantes.

Or, il y a $\binom{n}{k} = C_n^k$ manières de choisir k places parmi n. D'où on obtient $P(X = k) = C_n^k p^k q^{n-k}$. On dit que la variable aléatoire suit une loi binomiale de paramètres n et p. On note cette distribution $X \sim \beta(n, p)$. Cette distribution dépend des paramètres n et p.

Distributions multinomiale

La loi *multinomiale* s'applique à k évènements complémentaires de probabilités $p_1, p_2, \ldots, p_k$. Sur n tirages ou expériences. La probabilité que l'on observe x_i fois les évènements de probabilité p_i s'écrit :

$$P((X_1, \ldots, X_k) = (x_1, \ldots, x_k)) = \frac{n!}{x_1! \ldots, x_k!} p_1^{x_1} \ldots p_k^{x_k}.$$

La distribution multinomiale de la variable aléatoire $X = (X_1, \ldots, X_k)$ se caractérise par les $k+1$ paramètres $n, p_1, p_2, \ldots, p_k$ et se note $X \sim \mathcal{M}(n, p_1, p_2, \ldots, p_k)$.

Distributions Dirichlet

Soit X un vecteur aléatoire avec des élément $0 \leq x_i \leq 1$ et $\sum x_i = 1$.

$$\theta = (\alpha_1, \alpha_2, \ldots, \alpha_k) \in \Theta = \mathbb{R}_+^k,$$

où α_i sont des paramètres de concentration.

La distribution de *Dirichlet* est donnée par :

$$P(X, \alpha) = \frac{\Gamma(\sum_{i=1}^k \alpha_i)}{\prod_{i=1}^k \Gamma(\alpha_i)} \prod_{i=1}^k x_i^{\alpha_i - 1},$$

La fonction gamma (Γ) est une généralisation de la fonction factorielle aux réels strictement positifs. Pour $a > 0$, $\Gamma(a) = \int_0^{+\infty} \mathrm{e}^{-x} x^{a-1}\, dx$.

On a $\forall a \in \mathbb{N}^*, \Gamma(a) = \begin{cases} 1 & \text{si } a = 1, \\ (a-1)! & \text{sinon .} \end{cases}$

$\forall a \in]1, +\infty[, \Gamma(a) = (a-1)\Gamma(a-1)$.

La distribution de Dirichlet se caractérise par les paramètres α et se note $X \sim Dir(\alpha)$.

1.2 Représentation du texte

Les algorithmes d'apprentissage automatique sont développés pour être appliqués sur des donnés vectorielles. Cependant pour plusieurs types de données, cette représentation n'est pas disponible à l'état brut.

Dans le contexte de notre projet, les données textuelles doivent être pré-traitées avant de servir d'entrées pour un système d'apprentissage.

En effet, il existe plusieurs façons de représenter une entité textuelle. Dans la présente section, nous considérons la représentation sous forme de *sac de mots* (bag of words) et la représentation *word2vec*.

1.2.1 Représentation en sac à mots

La représentation en «sac de mots» ou modèle d'espace vectoriel (Vector Space Model, VSM) est introduit par [Salton et al., 1975].

L'idée principale consiste à représenter une entité textuelle notée d (pour document) sous forme d'un vecteur indexé par les mots (termes) qu'elle contient.

L'élément $tf\ (t_i, d)$ est la fréquence (le nombre d'occurrences) du terme t_i dans le document d. La nature du terme t_i est en général, le résultat des opérations linguistiques lors de la phase de sélection des attributs dans le processus du prétraitement (tokenisation, lematisation, filtrage des mots vides, ...).

Formellement, cette représentation peut être modélisée comme étant une projection d'un document d dans un espace de haute dimension :

$$d \mapsto \phi(d) = (tf(t_1, d), tf(t_2, d), \ldots, tf(t_n, d)) \in \mathbb{R}^n$$

Il est clair que, dans cette représentation, l'ordre des mots et les informations grammaticales sont ignorés, ainsi, il est impossible de reconstruire le document original à partir de sa représentation en «sac de mots».

Un problème avec l'utilisation de la fréquence des mots $(tf(t_i, d))$ est que les mots très fréquents tendent à dominer dans le document, mais peuvent ne pas contenir autant de contenu informationnel pour le modèle que des mots plus rares.

Une alternative consiste à redimensionner la fréquence des mots en fonction de leur fréquence d'apparition dans tous les documents, de sorte que les scores des mots fréquents comme « le, la, de, ... » qui sont également fréquents dans tous les documents soient pénalisés. Une telle approche est appelée tf-*idf* pour *terme frequency-inverse document frequency* qui est donnée formellement par l'Équation1.2

$$tf\text{-}idf\,(t_i, d) = tf(t_i, d) * idf(t_i, D), \tag{1.2}$$

où

$$idf(t_i, D) = \log \frac{|D|}{|\,\{d \in D | t_i \in d\}\,|}$$

avec

— $|D|$: nombre total des documents dans le corpus.

— $|\,\{d \in D | t_i \in d\}\,|$: nombre de documents où le terme t_i apparaît.

Le score $tf\text{-}idf$ a pour effet de mettre en évidence des mots distincts, contenant des informations utiles, dans un document donné.

Limitations de la représentation en « sac de mots »

Le modèle du sac de mots [Joachims, 1998] est très simple à comprendre et à mettre en oeuvre. Il a été utilisé avec un grand succès dans des problèmes de prédiction tels que la modélisation des langages et la classification des documents. Néanmoins, il souffre de certaines lacunes :

- Vecteurs éparses : la représentation en « sac de mots » engendre des vecteurs creux (la plupart de ses entrées sont égales à zéro). Ceci pent causer deux problèmes. Le premier est un problème calculatoire (complexité spatiale et temporelle). Le seconde est un problème informationnel, où le défi est d'exploiter peu d'informations dans un grand espace de représentation.
- Sémantique : la représentation en « sac de mots» ignore toute relation sémantique entre les mots. Ceci est dû à la suppression de l'ordre des mots qui écarte le contexte. Le contexte peut offrir beaucoup au modèle. À titre d'exemple les mots synonymes et homonymes.

1.2.2 Représentation word2vec

La représentation *word2vec* [Mikolov et al., 2013] repose sur le principe du contexte du terme. Les termes fréquemment trouvés approchés dans une collection de documents apparaîtront également approchés dans l'espace word2vec.

Word2vec est un moyen de convertir des termes en vecteurs, de telle manière que les similarités entre les termes puissent être découvert mathématiquement.

Word2vec prend en entrée un corpus de texte de taille importante et le vectorise à l'aide d'un réseau de neurones peu profond. Le résultat est une liste de terme (vocabulaire), chacun avec un vecteur correspondant. Les termes ayant une signification similaire se produisent dans l'espace de proximité. La mesure de similarité cosinus est utilisée.

Les termes peuvent être encodés sous forme de vecteurs en utilisant différents modèles. Dans [Mikolov et al., 2013], les auteurs ont examiné deux modèles existants, le modèle de langage de

réseau de neurones à rétroaction (Neural Net Language Model, NNLM) et le modèle de langage de réseau de nuerons récurrent (Recurrent Neural Net Language Modele, RNNLM).

De plus, ils proposent deux nouveaux modèles, un '*sac de mot continu* (Continuous Bag Of Word, CBOW) et un *Skip-gram continu*.

Dans leurs comparaisons, CBOW et Skip-gram ont obtenu de meilleurs résultats. Dans la suite, nous examinons le modèle Skip-gram car il a une relation avec la modélisation thématique *Lda2vec* que nous développons dans la Section 2.2.

Modèle Skip-gram

Le modèle Skip-gram est utilisé pour prédire les termes de contexte d'un terme donné (CBOW cherche à prédire un terme à partir de ses termes de contexte).

La Figure 1.1 montre l'architecture du modèle Skip-gram.

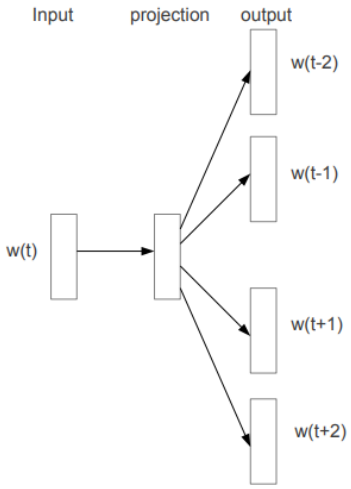

FIGURE 1.1 – Architecture du modèle Skip-gram de la représentation word2vec.

Le terme $w(t)$ est le vecteur d'entrée. Il existe une couche cachée qui effectue le produit scalaire entre la matrice de pondération et le vecteur d'entrée $w(t)$. Il est à signaler qu'il n'existe aucune fonction d'activation au niveau de la couche cachée. Le résultat du produit scalaire au niveau de la couche cachée est transmis à la couche de sortie. Cette dernière effectue le produit scalaire entre le vecteur de sortie de la couche cachée et la matrice de pondération de la couche de sortie. Enfin, la fonction d'activation softmax est appliquée pour calculer la probabilité que les termes apparaissent dans le contexte de $w(t)$ à un emplacement de contexte donnée.

Plus formellement :

- V représente le *vocabulaire* ou le *dictionnaire* des termes présents dans une collection de données.
- N dénote le nombre de neurones dans la couche cachée.
- La taille de la fenêtre est l'emplacement du contexte maximal auquel les termes doivent être prédits. Elle est dénotée par c.

- La *fenêtre du contexte* est le nombre de termes à prédire qui peuvent apparaître dans la plage d'un terme donnée. Elle est représentée par $k = 2 \times c$.
- La dimension du vecteur d'entrée est $|V|$. Chaque terme est encodé en utilisant le *one hot encoding*.
- La matrice de pondération pour la couche cachée (W) est de dimension $[|V|, N]$.
- Le vecteur de sortie de la couche cachée est $H[N]$.
- La matrice de pondération entre la couche cachée et celle de sortie ($W^{'}$) est de dimension $[N, |V|]$.
- Le produit scalaire entre $W^{'}$ et H donne un vecteur de sortie $U[|V|]$.

Après avoir définit les variables du système, décrivons, étape par étape, le processus de prédiction des termes de contexte pour un terme *cible* :

1. Les termes sont convertis en vecteurs en utilisant un one hot encoding. La dimension de ces vecteurs est $[1, |V|]$.
2. Le terme $w(t)$ est passé à la couche cachée composée de |V| neurones.
3. La couche cachée effectue le produit scalaire entre la matrice de pondération $W[|V|, N]$ et le vecteur d'entrée $w(t)$. Nous pouvons conclure que la (t) ème ligne de $W[|V|, N]$ sera la sortie $H[1, N]$.
4. Rappelons qu'aucune fonction d'activation n'est utilisée au niveau de la couche cachée, donc $H[1, K]$ sera transmis directement à la couche de sortie.
5. La couche de sortie applique le produit scalaire entre $H[1, N]$ et $W^{'}[H, |V|]$ et nous donne le vecteur U.
6. Pour trouver la probabilité de chaque vecteur, la fonction softmax intervient.
7. Le terme avec la probabilité la plus élevée est le résultat. Si le terme prédit pour une position de contexte est faux, nous utilisons la rétropropagation pour modifier les matrices de pondération W et $W^{'}$.

Ces étapes vont être exécutées pour chaque terme $w(t)$ présent dans le vocabulaire.

Nous clôturons cette section par l'énumération de quelques avantages et inconvénients du modèle skip-gram :

Avantages

1. Le modèle skip-grame est un apprentissage non supervisé et peut donc fonctionner avec des données non étiquetées
2. Elle nécessite deux matrices de pondération de dimension $[N, |V|]$ chacune au lieu de $[|V|, |V|]$. Généralement $N \ll |V|$.

Inconvénients

1. Trouver la meilleure valeur pour N et c n'est pas évident.

2. La complexité de calcul de la fonction *softmax* est très coûteuse.
3. Le temps d'apprentissage de l'algorithme est élevé.

Chapitre 2

Modélisation thématique

Dans le présent chapitre, nous introduisons le concept de la modélisation thématique (Topic Modeling), ses techniques et enfin nous clôturons avec quelques applications de la modélisation thématique.

Toute au long ce chapitre, nous procédons à une étude bibliographique de notre sujet.

2.1 Présentation de la modélisation thématique

L'approche à base de dictionnaires utilise la fréquence des termes pour l'extraction de la sémantique du texte. Cette approche marque sa limitation, elle suppose que chaque terme possède un et un seul sens. Une supposition plus raisonnable stipule que les termes peuvent avoir différentes significations en fonction de leurs contextes (apparition à coté d'autres termes).

La modélisation thématique est une classe des méthodes d'analyse de texte qui s'intéresse à l'analyse d'un sac (bag) de mots (au lieu de compter les termes individuellement) pour capturer le sens des termes en fonction de leurs contextes dans un langage naturel.

Autrement dit, la modélisation thématique est une forme de text mining employant des techniques d'apprentissage automatique supervisés et non supervisés pour identifier des motifs dans un corpus contenant une importante quantité de texte non structuré.

Elle regroupe les termes d'un tel corpus dans des clusters de termes, générant des thématiques (topics) en utilisant des processus de similarité.

À titre d'illustration, la Figure 2.1 montre un exemple de modélisation thématique appliquée sur un corpus d'articles depuis le journal Science [Blei, 2012].

Elle montre comment un petit morceau de texte d'un seul document est classifié en utilisant la modélisation thématique. Les cases colorées (à gauche) décrivent les thématiques identifiées par le modèle et les termes dans chaque case décrivent les termes les plus fréquents qui apparaissent dans chaque thématique. L'histogramme (à droite) décrit le mélange (mixture) des thématiques identifiées dans ce document particulier.

Il est important de noter que la modélisation thématique ne remplace pas l'interprétation humaine du texte. En fait, c'est un moyen de faire des suppositions sur la façon dont les termes

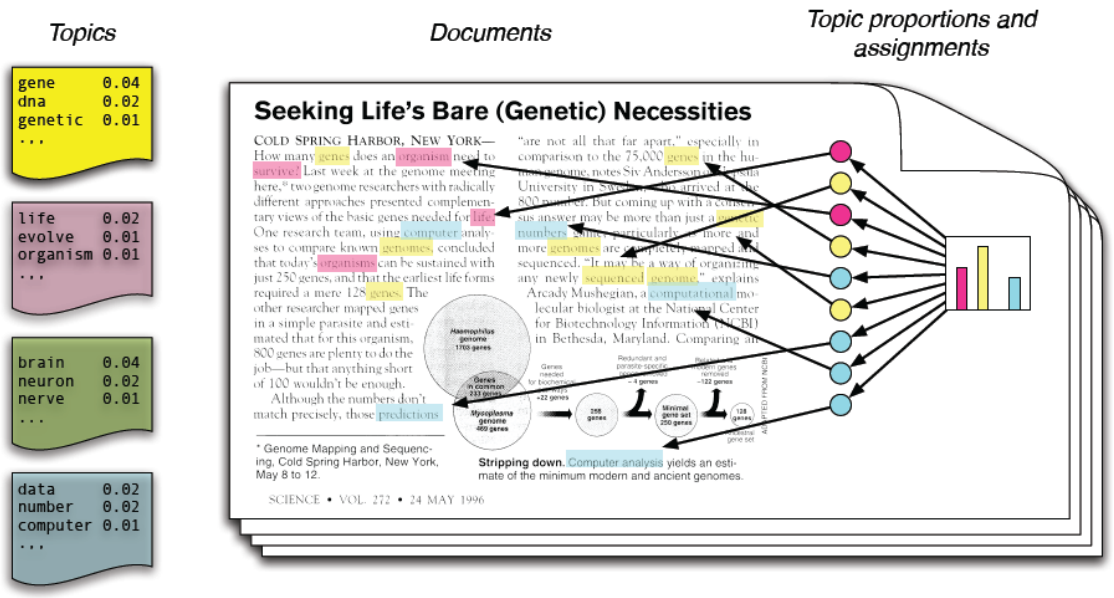

FIGURE 2.1 – Exemple de modélisation thématique [Blei, 2012].

co-occurrent dans différentes thématiques latentes en identifiant des modèles dans la façon dont il coexistent dans les documents.

2.2 Différentes approches de la modélisation thématique

Dans cette section, nous décrivons différentes approches de l'analyse thématique, à savoir, l'analyse sémantique latente (Latente Semantic Analysis, LSA), l'analyse sémantique latente probabiliste (Probabilistic Latent Sementic Analysis, PLSA) et l'allocation de Dirichlet latente (Latent Dirichlet Allocation, LDA).

2.2.1 Analyse Sémantique Latente

L'analyse sémantique latente (Latent Semantic Analysis, LSA) a été introduite pour la première fois par [Deerwester et al., 1990].

L'approche LSA est utilisée comme une méthode automatique d'indexation et de recherche. Elle s'appuie sur la découverte d'une structure sémantique des relations entre les termes et les documents.

La méthode LSA, est ensuite utilisée dans le domaine de la modélisation thématique. Elle s'appuie sur une analyse mathématique profonde capable de déduire correctement des relations profondes.

L'idée principale du LSA est qu'il existe un ensemble de contraintes mutuelles entre les documents. Ces contraintes déterminent la similarité entre les termes ou bien les ensembles des termes (documents). Cette méthode vise à mettre en évidence ces contraintes pour déduire les thématiques des documents.

Dans la première étape, la méthode LSA représente le corpus dans une matrice terme-document. Dans la littérature, les cases d'une telle matrice peuvent être renseignées soit par les fréquences des termes (tf) ou par les valeurs tf-idf (Voir Section 1.2.1).

Cependant la représentation matricielle terme-document présente plusieurs problèmes, liée essentiellement à la dimension élevée ainsi que le bruit causé par les termes.

Une solution consiste à faire recours à la méthode de décomposition en valeurs singulières (Singular Value Decomposition, SVD) [Strang, 2006].

Décomposition en valeurs singulières (SVD)

L'intérêt essentiel de l'utilisation de la méthode mathématique SVD par LSA est qu'elle prend en entrée une matrice termes-documents et la représente sous forme de composantes linéairement indépendantes.

Ces composantes, peuvent être interprétées comme vecteurs propre à partir des données brutes de la matrice terme-document. En effet, cette nouvelle représentation est considérée comme une réduction substantielle de la dimensionalité de la matrice d'origine.

Par ailleurs, SVD dispose d'un autre avantage autre que la réduction de la dimensionalité. Les documents concernant une thématique particulière deviennent plus similaires, même si les termes identiques n'apparaissent pas tous dans ces documents.

Comme nous l'avons mentionné précédemment, la matrice d'origine est décomposée en trois matrices. Selon l'équation (2.1) :

La méthode SVD décompose la matrice d'origine en trois matrices :

$$A = U_{m*m} S_{m*n} V_{n*n} \tag{2.1}$$

Où :

- A : est la matrice d'origine de taille $m * n$ par exemple m termes et n documents.
- U : est une matrice orthogonale de vecteurs de termes dont ses colonnes sont des vecteurs propres de AA^T.
- V : une matrice orthogonale de vecteurs de documents où ses colonnes sont des vecteurs propres de $A^T A$.
- S : une matrice diagonale contenant les valeurs propres.

À titre d'illustration, nous utilisons l'exemple utilisé dans l'article principal de LSA [Deerwester et al., 1990]. L'ensemble des documents est constitué de neuf titres de mémoires. Cinq mémoires sur l'interaction homme-machine (c1..c5) et les quatre autres sur la théorie des graphes (m1..m4). La Table (2.1) montre le corpus utilisé.

Nous représentons ces titres dans une matrice terme-document (Table 2.1).
Ensuite, nous appliquons la méthode SVD à la matrice A représentant la matrice terme-document (Table 2.2).

TABLE 2.1 – Corpus d'un exemple d'illustration de la méthode LSA.

C1	Human machine interjuce for Lab ABC computer applications
C2	A survey of user opinion of computer system response time
C3	The EPS user interface management system
C4	System and human system engineering testing of EPS
C5	Relation of user-perceived response time to error measuremen
M1	The generation of random, binary, unordered trees
M2	The intersection graph of paths in trees
M3	Graph minors IV : Widths of trees and well-quasi-ordering
M4	Graph minors : A survey

$$A = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Nous passons, ensuite, au calcul de la matrice gauche des vecteurs propres (U) :

Nous commençons par le calcule de AA^T.

TABLE 2.2 – Représentation des titres dans une matrice terme-document.

Terme	Document								
	C1	C2	C3	C4	C5	M1	M2	M3	M4
Human	1	0	0	1	0	0	0	0	0
Interface	1	0	1	0	0	0	0	0	0
Computer	1	1	0	0	0	0	0	0	0
User	0	1	1	0	1	0	0	0	0
System	0	1	1	2	0	0	0	0	0
Response	0	1	0	0	1	0	0	0	0
Time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
Survey	0	1	0	0	0	0	0	0	1
Trees	0	0	0	0	0	1	1	1	0
Graph	0	0	0	0	0	0	1	1	1
Minors	0	0	0	0	0	0	0	1	1

$$AA^T = \begin{pmatrix} 2 & 1 & 1 & 0 & 2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 3 & 2 & 2 & 2 & 1 & 1 & 0 & 0 & 0 \\ 2 & 1 & 1 & 2 & 6 & 1 & 1 & 3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 2 & 2 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 2 & 2 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 3 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 2 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 2 & 2 \end{pmatrix}$$

Les valeurs propres de AA^T, sont :

$\lambda_1 = 11.1615$, $\lambda_2 = 6.46024$, $\lambda_3 = 5.54105$, $\lambda_4 = 2.70449$,
$\lambda_5 = 2.26452$, $\lambda_6 = 1.70663$, $\lambda_7 = 0.715552$, $\lambda_8 = 0.313751$,
$\lambda_9 = 0.132261$, $\lambda_{10} = 0$, $\lambda_{11} = 0$, $\lambda_{12} = 0$

Nous rappelons que U est constituée des vecteurs propres liés aux valeurs propres ordonnées par ordre décroissant de gauche à droite :

$$U = \begin{matrix} 0.22 & 0.11 & 0.29 & -0.41 & -0.11 & -0.34 & 0.52 & 0.06 & -0.41 \\ 0.20 & -0.07 & 0.14 & -0.55 & 0.28 & 0.50 & -0.07 & -0.01 & -0.11 \\ 0.24 & 0.04 & -0.16 & -0.59 & -0.11 & -0.25 & -0.30 & 0.06 & 0.49 \\ 0.40 & 0.06 & -0.34 & 0.10 & 0.33 & 0.38 & 0.00 & 0.00 & 0.01 \\ 0.64 & -0.17 & 0.36 & 0.33 & -0.16 & -0.21 & -0.17 & 0.03 & 0.27 \\ 0.27 & 0.11 & -0.43 & 0.07 & 0.08 & -0.17 & 0.28 & -0.02 & -0.05 \\ 0.27 & 0.11 & -0.43 & 0.07 & 0.08 & -0.17 & 0.28 & -0.02 & -0.05 \\ 0.30 & -0.14 & 0.33 & 0.19 & 0.11 & 0.27 & 0.03 & -0.02 & -0.17 \\ 0.21 & 0.27 & -0.18 & -0.03 & -0.54 & 0.08 & -0.47 & -0.04 & -0.58 \\ 0.01 & 0.49 & 0.23 & 0.03 & 0.59 & -0.39 & -0.29 & 0.25 & -0.23 \\ 0.04 & 0.62 & 0.22 & 0.00 & -0.07 & 0.11 & 0.16 & -0.68 & 0.18 \\ 0.03 & 0.45 & 0.14 & -0.01 & -0.30 & 0.28 & 0.34 & 0.68 & 0.18 \end{matrix}$$

De la même manière nous construisons la matrice V^T (matrice droite des vecteurs propres) associée à A^TA.

Les valeurs propres de A^TA sont :

$\lambda_1 = 11.1615$, $\lambda_2 = 6.46024$, $\lambda_3 = 5.54105$,
$\lambda_4 = 2.70449$, $\lambda_5 = 2.26452$, $\lambda_6 = 1.70663$,
$\lambda_7 = 0.715552$, $\lambda_8 = 0.313751$, $\lambda_9 = 0.132261$,

$$V^T = \begin{matrix} 0.20 & -0.06 & 0.11 & -0.95 & 0.05 & -0.08 & 0.18 & -0.01 & -0.06 \\ 0.61 & 0.17 & -0.50 & -0.03 & -0.21 & -0.26 & -0.43 & 0.05 & 0.24 \\ 0.46 & -0.03 & 0.21 & 0.04 & 0.38 & 0.72 & -0.24 & 0.01 & 0.02 \\ 0.54 & -0.23 & 0.57 & 0.27 & -0.21 & -0.37 & 0.26 & -0.02 & -0.08 \\ 0.28 & 0.11 & -0.51 & 0.15 & 0.33 & 0.03 & 0.67 & -0.06 & -0.26 \\ 0.00 & 0.19 & 0.10 & 0.02 & 0.39 & -0.30 & -0.34 & 0.45 & -0.62 \\ 0.01 & 0.44 & 0.19 & 0.02 & 0.35 & -0.21 & -0.15 & -0.76 & 0.02 \\ 0.02 & 0.62 & 0.25 & 0.01 & 0.15 & 0.00 & 0.25 & 0.45 & 0.52 \\ 0.08 & 0.53 & 0.08 & -0.03 & -0.60 & 0.36 & -0.04 & -0.07 & -0.45 \end{matrix}$$

La matrice S qui contient les racines carrées des valeurs ordonnées du plus grand au petit le long de sa diagonale.

$$S = \begin{matrix} 3.34 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2.54 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2.35 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.64 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.50 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.31 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.85 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.56 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.36 \end{matrix}$$

Après avoir multiplié les trois matrices, nous obtenons une nouvelle matrice qui est très similaire à celle d'origine. Le but de l'utilisation de SVD est de réduire les dimensions, et pour cela, il est important de se débarrasser de certaines valeurs propres de la matrice S, ayant des petites valeurs (représentant le bruit). Il en découle d'éliminer les vecteurs propres associes dans

les matrices U et V, tout en maintenant l'énergie des valeurs propres (Équation 2.2) entre 80% et 90%. L'énergie est calculée comme suit :

$$E = \sum \lambda_i^2 \tag{2.2}$$

La relation de la nouvelle matrice impliquant les valeurs propres est la suivante (Équation 2.3) :

$$\widehat{A} = U_{m*K} S_{K*K} V_{K*n} \tag{2.3}$$

pour cet exemple, $K = 2$:

$$\begin{matrix} 0.22 & -0.11 & 3.34 & & 0.20 & 0.61 & 0.46 & 0.54 & 0.28 & 0.00 & 0.02 & 0.02 & 0.08 \\ 0.20 & -0.07 & & 2.45 & -0.06 & 0.17 & -0.13 & -0.23 & 0.11 & 0.19 & 0.44 & 0.62 & 0.53 \end{matrix}$$

$$\begin{matrix} 0.24 & 0.04 \\ 0.40 & 0.06 \\ 0.64 & -0.17 \\ 0.27 & 0.11 \\ 0.27 & 0.11 \\ 0.30 & -0.14 \\ 0.21 & 0.27 \\ 0.01 & 0.49 \\ 0.04 & 0.62 \\ 0.03 & 0.45 \end{matrix}$$

$$\widehat{A} = \begin{matrix} 0.16 & 0.40 & 0.38 & 0.47 & 0.18 & -0.05 & -0.12 & -0.16 & -0.09 \\ 0.14 & 0.37 & 0.33 & 0.40 & 0.16 & -0.03 & -0.07 & -0.10 & -0.04 \\ 0.15 & 0.51 & 0.36 & 0.41 & 0.24 & 0.02 & 0.06 & 0.09 & 0.12 \\ 0.26 & 0.84 & 0.61 & 0.70 & 0.39 & 0.03 & 0.08 & 0.12 & 0.19 \\ 0.45 & 1.23 & 1.05 & 1.27 & 0.56 & -0.07 & -0.15 & -0.21 & -0.05 \\ 0.16 & 0.58 & 0.38 & 0.42 & 0.28 & 0.06 & 0.13 & 0.19 & 0.22 \\ 0.16 & 0.58 & 0.38 & 0.42 & 0.28 & 0.06 & 0.13 & 0.19 & 0.22 \\ 0.22 & 0.55 & 0.51 & 0.63 & 0.24 & -0.07 & -0.14 & -0.20 & -0.11 \\ 0.10 & 0.53 & 0.23 & 0.21 & 0.27 & 0.14 & 0.31 & 0.44 & 0.42 \\ -0.06 & 0.23 & -0.14 & -0.27 & 0.14 & 0.24 & 0.55 & 0.77 & 0.66 \\ -0.06 & 0.34 & -0.15 & -0.30 & 0.20 & 0.31 & 0.69 & 0.98 & 0.85 \\ -0.04 & 0.25 & -0.10 & -0.21 & 0.15 & 0.22 & 0.50 & 0.71 & 0.62 \end{matrix}$$

Enfin, la complexité du calcul du SVD étant $O(\min(mn^2, m^2n))$, ceci est du à la multiplication matricielle. Où m est la taille du corpus et n est la taille du vocabulaire.

LSA avec des méthodes alternatives

Nous pouvons constater que la matrice $\widehat{A}$ comprend des valeurs négatives qui n'ont pas de signification dans la représentation terme-document.

Une alternative consiste à utiliser la technique de factorisation par matrices non négatives (Non négative Matrix Factorisation, NMF) au lieu du SVD, car elle garantit qu'aucune valeur négative n'apparaît dans la matrice finale.

La NMF est une technique proposé par [Lee and Seung, 1999] dont le principe peut être décrit comme suit :

Soit A une matrice $(m \times n)$ contenant des valeurs non négatives et aucune ligne ou colonne contenant uniquement des zéro ; r est un entier relativement petit qui est choisi entre m et n.

l'idée consiste à réaliser un produit matricielle entre deux matrices $W_{m\times r}$ et $H_{r\times n}$ de telle sorte que leur produit se rapproche de

$$\widehat{A} = WH$$

Les deux méthodes (NMF et SVD) ont été appliquées sur l'ensemble de données MEDLINE. Dans le domaine de la recherche d'information [Peter and Kp, 2009], les résultats des

expérimentation conduites ont révélé qu'il n'existe pas une déférence significative entre les deux méthodes en matière de précision et de rappel. Cependant, NMF a un avantage sur SVD, car elle donne une représentation naturelle avec l'absence des éléments négatifs qui peuvent être utiles dans l'application de LSA.

Critique de LSA

L'avantage principal de la méthode LSA est qu'elle s'appuie sur un formalisme mathématique solide. Cependant, elle a encore quelque inconvénients [Hofmann, 1999] :

- La complexité temporelle de LSA n'est pas appropriée pour le big data.
- La méthode LSA ne donne pas de résultats précis avec de petits documents car elle nécessite un grand nombre de termes.

2.2.2 Analyse sémantique latente probabiliste

Bien que LSA a été appliquée avec succès dans certains domaines, y compris l'indexation automatique, elle présente un nombre de déficits, principalement dus à sa base statistique insatisfaisante. L'objectif principal de l'analyse sémantique latente probabiliste (Probabilistic Latent Semantic Analysis, PLSA) [Hofmann, 1999] est d'assurer une base statistique solide à LSA. En effet, l'approche PLSA est basée sur les principes de probabilité et définit, ainsi, un modèle génératif approprié des données.

Le point de départ du PLSA est un modèle statistique appelé modèle d'aspect [Hofmann, 2001]. Le modèle d'aspect est un modèle de variables latentes pour les données de co-occurrence qui associe une variable de classe non observée.

Définissons formellement les variables mises en oeuvre par PLSA : l'ensemble des documents $D = d_1, d_2, \ldots, d_N$, N étant le nombre de documents et d_i dénote le $i$ème document dans l'ensemble D.

L'ensemble des termes (mots) $W = w_1, w_2, \ldots, w_M$, M étant la taille du vocabulaire et w_i dénote le $i$ème terme dans le vocabulaire W. Notons que l'ensemble W est traité comme un sac de mots. Cela implique qu'il n'existe pas un ordre particulier pour assigner l'indice i. Enfin, l'ensemble des thématiques (variables latentes ou cachées) $Z = z_1, z_2, ..., z_k$, le nombre k est un paramètre à spécifier.

Comme mentionné précédemment, les thématiques sont des variables cachées. La seule chose que nous manipulons (observons) concrètement, ce sont l'ensemble des termes et l'ensemble des documents. L'objectif du PLSA est de mettre en relation des variables cachées avec les variables observées.

La façon dont nous associons z à (d, w) consiste à décrire un processus génératif où il est possible de générer un document, puis une thématique puis un terme. Ceci peut être décrit selon le schéma suivant :

1. Sélectionner un document d_i avec une probabilité $P(d_i)$.

2. Pour chaque document d_i, choisir une classe latente z_k à partir d'une distribution conditionnelle $P(z_k|d_i)$.
3. Pour chaque variable cachée z_k, générer un terme w_j avec une probabilité conditionnelle $P(w_j|z_k)$.

Le développement mathématique qui suit repose sur les deux hypothèses suivantes :

Hypothèse 1 : Sac de mots : nous avons déjà annoncé que dans ce type de représentation du texte, l'ordre n'importe pas. Plus précisément, la variable conjointe (d, w) est échantillonnée indépendamment (Équation 2.4).

$$P(D, W) = \prod_{(d,w)} P(d, w) \tag{2.4}$$

Hypothèse 2 : Indépendance conditionnelle : les termes et les documents sont conditionnellement indépendants (Équation 2.5).

$$P(W, d|Z) = P(W|Z) * P(d|Z) \tag{2.5}$$

Le modèle décrit ci-dessus (associer z à (d, w)) peut être spécifié comme suit :

$$P(d, w) = P(d)P(w|d) \tag{2.6}$$

où

$$P(w|d) = \sum_{z \in Z} P(w, z|d) = \sum_{z \in Z} P(w|d, z)P(z|d) \tag{2.7}$$

En utilisant l'indépendance conditionnelle :

$$P(w|d) = \sum_{z \in Z} P(w|z)P(z|d) \tag{2.8}$$

En utilisant la règle de Bayes :

$$P(d, w) = \sum_{z \in Z} P(z)P(d|z)P(w|z) \tag{2.9}$$

Il est claire que l'Équation (2.9) établit un lien entre la variable latente z et les variables observées (d, w).

Les paramètres du modèle décrit par l'Équation (2.9) sont estimés en maximisant la vraisemblance L donnée par l'Équation (2.10)

$$L = \sum_{d \in D} \sum_{w \in W} n(d, w) \log P(d, w) \tag{2.10}$$

où $n(d, w)$ dénote la fréquence du terme, c'est à dire le nombre de fois où w apparaît dans

le document d.

La procédure standard pour l'estimation du maximum de vraisemblance est l'algorithme Expectation-Maximisation (EM) [Dempster et al., 1977].

L'algorithme EM alterne deux étapes :

1. Une étape d'espérance (Expectation, E) où les probabilités a posteriori $P(z|d,w)$ sont calculées pour les variables latente z, sur la base des estimations courantes des paramètres.
2. Une étape de maximisation (M), où les paramètres courantes sont mis à jour sachant les probabilités a posteriori déterminées dans la phase (E).

Dans la phase d'espérance (E), on calcule la probabilité qu'un terme w dans un document particulier ou texte d soit déterminé par la variable latente z (Équation 2.11).

$$P(z|d,w) = \frac{P(z)P(d|z)P(w|z)}{\sum_{z' \in Z} P(z')P(d|z')P(w|z')} \tag{2.11}$$

Par des calcules standards on peut obtenir les équations de re-estimation de l'étape (M) :

$$P(w|z) = \frac{\sum_{d \in D} n(d,w)P(z|d,w)}{\sum_{d \in D, w' \in W} n(d,w')P(z|d,w')} \tag{2.12}$$

$$P(d|z) = \frac{\sum_{w \in W} n(d,w)P(z|d,w)}{\sum_{d' \in D, w \in W} n(d',w)P(z|d',w)} \tag{2.13}$$

$$P(w|z) = \frac{1}{R} \sum_{d \in D, w \in W} n(d,w)P(z|d,w) \tag{2.14}$$

avec

$$R = \sum_{d \in D, w \in W} n(d,w) \tag{2.15}$$

Les étapes (E) (Équation 2.11) et (M) (Équation 2.12-2.15) s'altèrent pour définir une procédure convergente qui s'approche d'un maximum local de la vraisemblance de l'Équation (2.10).

En fin, le modèle PLSA représente une amélioration remarquable du modèle LSA, néanmoins il comporte certains faiblesses :

- Le nombre de paramètres du modèle soit proportionnel avec la taille des documents et par conséquent, la taille du vocabulaire.
- La difficulté d'associer à un nouveau document pour le modèle PLSA, une distribution sur les mélanges de thèmes.

2.2.3 Allocation de Dirichlet latente

La méthode PLSA fonctionne bien comme une méthode totalement non supervisée pour l'analyse thématique des données textuelles. En effet, elle ne nécessite aucune intervention manuelle. Ceci peut être considéré comme un avantage dans le sens de la minimisation de l'effort humain, où la découverte des thématiques repose uniquement sur les caractéristiques des données sans considération de connaissances supplémentaires préalables sur les thématiques.

Dans la pratique, de telles connaissances supplémentaires sont souvent disponibles. Parfois, même les applications imposent une préférence particulière pour les thématiques à analyser.

Par conséquent, il est utile, voire nécessaire, d'imposer des connaissances préalable des paramètres à estimer afin que ces derniers soient conformes aux connaissances antérieures [Zhai and Massung, 2016].

Toutes ces connaissances préalables peuvent être incorporées dans PLSA en utilisant différents techniques, parmi lesquelles l'allocation de Dirichlet latente (Latent Dirichlet Allocation, LDA) [Blei et al., 2003].

La méthode LDA repose sur une idée simple qui suppose qu'un ensemble déterminé de thématiques est décrit à l'avance. Le modèle tient compte seulement des termes, caractéristiques observables, qui apparaissent dans les documents.

Autrement dit, la distribution de la couverture thématique (une distribution multinomiale) pour chaque document est supposée être tirée d'une distribution de Dirichlet apriori (car elle rend compte des 'croyances' de l'utilisateur avant l'observation des données), qui définit une distribution sur tout l'espace des paramètres d'une distribution multinominale, c'est à dire un vecteur de probabilités des thématiques.

De même, toutes les distributions de termes représentant les thématiques latentes dans une collection de texte sont également supposées être tirées d'une distribution de Dirichlet.

Dans PLSA, la distribution de la couverture thématique et les distributions de termes sont supposés être des paramètres (inconnus) dans le modèle, alors qu'en LDA, elles ne sont plus des paramètres du modèle puisqu'elles sont supposées être tirées des distributions de Dirichlet à priori correspondantes.

Par la suite, une fois que les distributions de termes pour la collection complète et celle de la couverture thématique pour un document soit échantillonnées, le reste du processus de génération des termes dans le document est exactement le même que PLSA.

Processus génératif

La Figure 2.2 illustre le modèle LDA. Dans un modèle génératif, les observations sont générées par des variables latentes. Dans LDA, compte tenu des termes, les thématiques latentes sont découvertes. Cependant, en tant que modèle génératif, on peut penser a LDA comme un modèle générant les thématiques puis les termes pour un certain document.

Dans la Figure 2.2, nous notons que α fixe une distribution particulière des thématiques θ.

Il existe une distribution pour chaque document. Pour un document donnée, lorsque nous choisissons une thématique dans cette distribution, nous sommes confortés à une distribution de termes β pour cette thématique. Ces distributions de termes sont déterminées par η.

La distribution des thématique θ et la distribution des termes β sont respectivement crées à partir de α et η. Ces derniers sont appelés priors de Dirichlet. Une valeur faible de α implique qu'un document peut avoir moins de thématiques dominantes. De même une valeur faible (ou élevée) de η signifie qu'une thématique a quelques (ou plusieurs) termes dominants.

L'algorithme d'apprentissage LDA commence par une étape d'initialisation qui consiste à attribuer une thématique à chaque terme de chaque document selon une distribution de Dirichlet avec un paramètre α (Équation 2.16).

$$\theta_i \sim Dir(\alpha), avec\, i \in \{1, \ldots, M\} \tag{2.16}$$

À titre d'exemple, dans un modèle à deux thématiques $\theta_i = [0.4, 0.6]$ signifie que le document d_i devrait avoir 40% de la thématique 1 et 60% de la thématique 2.

Cette étape d'initialisation se termine par un modèle initial qui est très peu vraisemblable du moment qu'il est généré aléatoirement.

La phase d'apprentissage tente d'améliorer le modèle thématique initial. Pour chaque document et pour chaque terme dans ce document on met à jour la thématique auquel il est lié Ceci se réalise, en calculant deux quantités pour chaque thématique :

- sélectionner une thématique $z_i \sim \mathcal{M}(\theta)$
- sélectionner un terme $w_i \sim \mathcal{M}(\beta)$

À l'issue de ces deux étapes, on choisit la nouvelle thématique. Ceci correspond à la probabilité que la thématique z_i génère le terme w_i d'un document d.

Le processus d'apprentissage se termine après un certains nombre d'itérations suffisant pour que les assignations se stabilisent.

Le résultat de cet algorithme d'apprentissage consiste à générer un mélange de thématiques présent dans chaque document ainsi que les termes associés à chaque thématique.

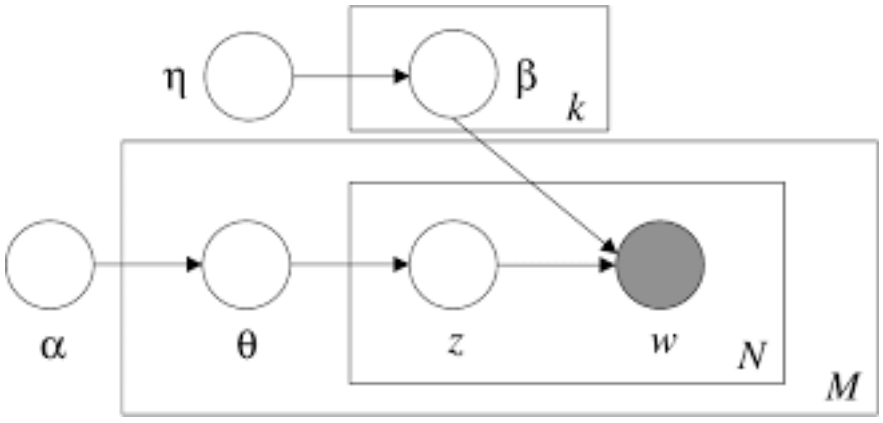

FIGURE 2.2 – Modéle graphique LDA

2.2.4 LDA2VEC

Il est connu que l'extraction du sens du texte est plus importante à tous les niveaux (terme, paragraphe, document). Au niveau du document, nous savons comment représenter le texte sous forme de mélanges de thèmatiques. Au niveau des termes, nous utilisons généralement quelque chose comme word2vec pour obtenir des représentations vectorielles. *Lda2vec* [Moody, 2016] est une extension de word2vec et de LDA qui apprend conjointement les vecteurs de termes, de documents et de thèmatiques.

Lda2vec s'appuie spécifiquement sur le modèle skip-gram de word2vec (Voir Section 1.2.2) pour générer des vecteurs de termes, il s'agit essentiellement d'un réseau de neurones qui apprend l'incorporation d'un terme en essayant d'utiliser le terme d'entrée pour prédire les termes de contexte.

Avec lda2vec, au lieu d'utiliser le vecteur de terme directement pour prédire les termes de contexte, nous utilisons un *vecteur de contexte* pour faire les prédictions. Ce vecteur de contexte est créé comme la somme de deux autres vecteurs : le *vecteur de terme* et le *vecteur de document*.

Le vecteur de mot est généré par le modèle skip-gram alors que le vecteur document est une combinaison pondérée de deux autres composants :

- Le vecteur de poids du document composé de poids de chaque thématique. De tels poids seront ensuite transformés en pourcentages de chaque thématique du document.
- La matrice des thématiques, représentant chaque thématique et son intégration vectorielle correspondante

Par la suite, le vecteur de document et le vecteur de mot génèrent des *vecteurs de contexte* pour chaque mot du document. De cette manière, lda2vec assure des représentations de mots, de contexte de mots, de thématiques et de documents.

2.3 Domaines d'applications

Dans cette section, nous nous concentrons sur les applications de modélisation thématiques pour résoudre des problèmes traditionnels. Nous passons en revue leur utilisation réussite par les chercheurs pour aider à comprendre des documents historiques, des publications scientifique, des textes politiques, ...

2.3.1 Comprendre les publications scientifiques

Les documents scientifiques utilisent un vocabulaire spécialisé définissant ainsi les domaines d'études. Par conséquence, les capacités de la modélisation thématique peuvent être utilisées dans la compréhension des publications scientifique. Cependant l'aspect innovant des documents scientifiques rend l'analyse des collections de documents scientifiques à la fois difficile et intéressante.

De plus, la compréhension de la publication scientifique est importante pour les agences de financent, les législateurs et le public.

Dans la Table 2.3, nous décrivons deux travaux qui s'intéressent à la compréhension de la publication scientifique.

TABLE 2.3 – Compréhension de la publication scientifique.

Référence	Intitulé / méthode	Description
[Griffiths and Steyvers, 2004]	Finding scientific topics / LDA	L'étude se focalise sur la compréhension des domaines d'études scientifiques. Elle utilise les résumés à partir des proceedings of the national academy of science (PNAS). Les résultats ont été satisfaisants car il a pu identifier les sujets scientifiques qui se distinguaient par des contextes et une terminologie scientifique bien connus.
[Talley et al., 2011]	Database of NIH grants using machine-learned categories and graphical clustering / LDA	NIH (the American National Institutes of Heath) est la première agence de financement américaine pour la recherche biologique et sanitaire. Le but de cette étude est de préparer une carte de financement de la recherche scientifique au profit du (NIH). Ce financement est important et difficile, ce qui a conduit à l'utilisation de la modélisation thématique pour aider à comprendre et financer la recherche scientifique en fonction de sa pertinence. Le corpus de cette étude contient des recherches scientifiques biologiques soumises par le NIH. L'application de la modélisation thématique à la collection de NIH n'était pas claire car le modèle nécessitait une intervention manuelle.

2.3.2 Recherche d'information Ad-hoc

L'analyse thématique analyse les documents au niveau sémantique. Ainsi elle offre une plate-forme unique et intéressante pour la modélisation des relations entre les termes et entre les termes et les documents [Deerwester et al., 1990, Hofmann, 1999].

Pour cette raison, l'analyse thématique a été appliquée avec succès dans le lissage des modèles de langage, expansion de requête et dans la recherche spécialisée.

Lissage dans les modèles de langage

Les modèles de langage créés ne peuvent reconnaître que des phrases qui apparaissent toutes en n grammes. Ce sont donc des modèles très limités et afin de les améliorer, nous assouplissons cette attribution systématique à probabilités nulles des termes ou des chaînes de termes à l'origine du problème. La procédure d'attribution d'une probabilité non nulle de ces éléments est appelée lissage. Le lissage peut également être considéré comme un moyen pour éviter de surentraînement du modèle sur un groupe et de donner au modèle une plus grande généralisation. La Table (2.4) résume deux travaux pour le lissage dans les modèles de langage utilisant la modélisation thématique.

TABLE 2.4 – Recherche d'information Ad-hoc : lissage dans les modèles de langages

Référence	Intitulé / méthode	Description
[Vosecky et al., 2014].	Collaborative personalized Twitter search with topic-language models / LDA	Explorer la méthode LDA pour les modèles de langage appliqué la recherche twitter. Les expérimentations ont montré que les performances de la plateforme sont comparables aux méthodes de base de l'état de l'art.
[Lu et al., 2011].	Investigating task performance of probabilistic topic models : an empirical study of PLSA and LDA / LDA et PLSA	Le papier étudie l'impact de quelques questions critiques (choix entre des modèles en compétition multiple maximum local,. . .) sur les performances des tâches utilisant la modélisation thématique (LDA et PLSA). Le papier implique trois tâches représentatives du text mining, à savoir : clustering des documents, catégorisation du texte et recherche ad-hoc. L'analyse expérimentale a fourni une compréhension profonde de l'analyse thématique et de nombreuse informations utiles sur la façon d'optimiser les performances des modèles thématiques pour ces taches typiques.

Modélisation thématique dans l'expansion de requêtes

La modélisation thématique est largement utilisée dans l'expansion de requêtes car elle exploite les relations sémantiques entre les termes à la place des appariements directes entre les termes [Zeng et al., 2012].Dans cettDans cette section, nous nous concentrons sur les applications de modélisation thématiques pour résoudre des problèmes traditionnels. Nous passons en revue leur utilisation réussite par les chercheurs pour aider à comprendre des documents historiques, des publications scientifique, des textes politiques, ...

e section, nous nous concentrons sur les applications de modélisation thématiques pour résoudre des problèmes traditionnels. Nous passons en revue leur utilisation réussite par les chercheurs pour aider à comprendre des documents historiques, des publications scientifique, des textes politiques, ...

La Table (2.5), résume certains travaux sur l'utilisation de la modélisation thématique dans l'expansion de requêtes.

TABLE 2.5 – Recherche d'information Ad-hoc : l'expansion de requête

Référence	Intitulé / méthode	Description
[Yi and Allan, 2009]	A comparative study of utilizing topic models for information retrieval /	L'étude explore l'utilité des différents types de la modélisation thématique dans la recherche d'information, en particulier, l'expansion de requête. Elle montre l'utilité de l'extraction des termes directement à partir des thèmes au lieu d'explorer l'ensemble du corpus. L'étude a montré que l'utilisation des modèles thématique améliore les performances de l'expansion de requête.
[Park and Ramamohanarao, 2009]	The sensitivity of latent dirichlet allocation for information retrieval / LDA	Dans cet article, les auteurs examinent la sensibilité de LDA par rapport au paramètre Dirichlet lorsqu'il est utilisé pour la recherche d'information (expansion de requêtes). Ils comparent les temps de calcul de modèle thématique, les exigences de stockage et la précision de récupération du LDA ajusté contre LDA avec un paramètre de Dirichlet uniforme. Les résultats obtenus montrent qu'il n'y a aucun avantage significatif à utiliser le LDA ajusté par rapport au LDA avec un paramètre de Dirichlet constant. Ceci montre que le LDA est insensible pour l'expansion de requêtes.

Recherche personnalisée

La recherche d'information personnalisée est une discipline centrée utilisateur. Autrement dit, ce type de systèmes de recherche d'information se focalise sur un utilisateur spécifique en vue de s'adapter à son contexte. La Table (2.6) présente quelques études traitant ce sujet en faisant référence à la modélisation thématique.

TABLE 2.6 – Recherche d'information Ad-hoc : Recherche personnalisée

Référence	Intitulé / méthode	Description
[Song et al., 2010]	Bridging topic modeling and personalized search / PLSA	L'idée est très similaire au lissage des modèles de langage de requête par des modèles de rubrique. Le résultat de cette méthode est de diriger l'utilisateur vers des documents qui correspondent aux intérêts de l'utilisateur.
[Dou et al., 2007]	A large-scale evaluation and analysis of personalized search strategies / LDA	Le but de l'article consiste à régler le problème de l'efficacité des différentes requêtes pour différents utilisateurs et dans différents contextes. Ce système diffère des systèmes traditionnels qui reposaient sur des requêtes uniquement, quel que soit le contexte : avec Internet, les mêmes mots sont utilisés dans des contextes très différents. Dans cet exemple, les chercheurs ont également cherché à comprendre les mots et le contexte, ce qui aide à comprendre les utilisateurs et leurs préférences, et ils ont finalement atteint des résultats très acceptables. les stratégies de personnalisation simples basées sur les clics fonctionnent de manière cohérente et considérable. les stratégies basées sur les profils sont instables dans cette expérience. les contextes à long et à court terme sont très importants pour améliorer les performances.

2.3.3 Analyse des documents historiques

La modélisation thématique joue un rôle important dans l'analyse des documents historiques. Ces derniers nécessitent beaucoup d'efforts et de temps pour les analyser. Une solution consiste à utiliser la modélisation thématique pour les analyser avec un minimum d'effort et de temps. La modélisation thématique joue le rôle d'un historien. À travers ses documents historiques, il découvrira des sujets historiques importants, la diversité culturelle, et comment vivait l'homme ancien !

En outre, la modélisation thématique nous aide à découvrir des faits historiques inattendus qui étaient auparavant inconnus.

Le temps est une variable critique dans l'étude des documents historiques. C'est une composante spécifique de la recherche historique. Mais ici, nous parlons de longues périodes qui peuvent être des décennies, voire des siècles. Par conséquent, de nombreux exemples que nous

mentionnons organisent les documents le long d'un axe temporel.

Dans la suite, nous nous intéressons d'abord, aux journaux, ensuite, nous considérons d'autres formes de documents historiques tels que les annales et les agendas.

2.3.4 Journaux

Les journaux historiques sont relativement proches aux journaux modernes qui sont des cas plus familier à la modélisation thématique (Table 2.8).

2.3.5 Documents historiques

Il existe d'autres types de documents historiques tels que les mémoire historiques décrivant la vie ancienne qui sont considérés comme un défi particulier. Nous tirons de ce type mon étude de cas dans différentes langues (Table 2.8).

TABLE 2.7 – Modélisation thématique sur les journaux

Référence	Intitulé/ méthode	Description
[Newman and Block, 2006a]	Probabilistic topic decomposition of an eighteenth-century american newspaper / LSA, PLSA, K-means	l'étude présente un exemple de modélisation thématique sur des journaux historiques appliqué à une collection d'articles de « the Pennsylvanie Gazette » depuis 1728 à 1800. Cette collection d'articles comprend 25 millions mots couvrant la vie quotidienne de plusieurs générations, avant, pendant et après la fondation des Etats-Unis d'Amérique. l'étude a dévoilé essentiellement les points suivant : une augmentation significative du débat politique dans la période post-révolution. • Un sujet lié aux descriptions de tissus est apparu dans les années 1750, mais décliné par la suite. • LSA fonctionne bien dans un espace de faible dimension, mais mal dans un espace de haute dimension, ce qui a conduit à des résultats insatisfaisants dans cet exemple. • K-means est plus similaire à PLSA et réussit mieux à trouver des étiquetés identifiables. Mais cela produit des combinaisons similaires avec de petites variations. Donc, le modèle k-means n'est pas assez expressif. • PLSA offre une flexibilité de modélisation et fournit des résultats interprétables.
[Yang et al., 2011]	Topic modeling on historical newspapers / LDA	l'étude modélise une collection de journaux historiques du Texas allant de 1829 à 2008. Le but de l'étude est de découvrir les intérêts des résidents du Texas au cours des 19 et 20 siècles, et de savoir plus sur des thèmes spécifiques, pré spécifiés tel que le Cotton. • L'étude a abouti aux résultats attendus et autres inattendus, comme la date d'établissement de la bataille de San Jacinto, le dernier conflit de la révolution texane qui a conduit à la sécession du Mexique.

TABLE 2.8 – Modélisation thématique sur les documents historique

Référence	Intitulé/ méthode	Description
[Erlin, 2017]	Topic modeling, epistemology, and the English and German novel. / LDA	L'étude recherche des travaux liés à l'épistémologie dans une collection de romans en anglais et en allemand. Les auteurs ont créé des modèles liés à des mots-clés spécifiques à chaque langue, tels que ceux utilisés pour la recherche d'information. Cette approche peut réussir, mais elle ne garantit pas que les thèmes pertinents soient trouvés.
[Miller, 2013]	Rebellion, crime and violence in qing china, 17221911 : A topic modeling approach / LDA	L'article utilise des disques de recherche chinois pour la signification de «zei» ou «bandits» sous la dynastie Qing en Chine (1644-1912). Le mot désigne un comportement hostile. la modélisation thématique utilise des informations contextuelles pour atteindre l'objectif. L'application de modélisation thématique LDA en chinois souligne l'importance du tokenisation et montre que LDA est très prometteuse.

Chapitre 3

Implémentation

3.1 Introduction

Dans ce chapitre, nous présentons une étude expérimentale. Nous voulons à partir de cette étude comparer deux méthodes bien connues dans le domaine de la modélisation thématique, à savoir, LSA et LDA. Nous mettons en œuvre chaque méthode et l'évaluons selon des critères spécifiques tels que le temps d'exécution et la cohérence thématique.

Les raisons principales de notre choix de LDA et LSA sont que LSA est considérée comme la première méthode suggérée dans le domaine de la modélisation thématique, car elle se distingue par sa simplicité. Nous choisissons LDA car c'est la méthode très souvent utilisée dans la littérature. Nous appliquerons cette étude à une collection de publications scientifiques de la conférence NIPS (Neural Information Processing Systems). Les publications scientifiques sont un type de document unique et se distinguent par une terminologie spéciale, ce qui rend notre tâche difficile par rapport à d'autres documents.

3.2 Environnement

Dans cette section, nous présentons le software et le hardware de l'environnement de développement de notre étude empirique.

3.2.1 Python

Pour implémenter notre système, le langage de programmation Python est le plus approprié, car il a fait de grands développements en programmation et est considéré comme un langage efficace, robuste, standard et facile à apprendre. De plus, Python peut bien gérer les données texte, grâce aux bibliothèques populaires telles que NLTK et Gensim qui aident au traitement du langage naturel, à la récupération d'informations et l'analyse thématique.

3.2.2 Gensim

Gensim est une bibliothèque Python pour l'analyse thématique et l'indexation. Cette bibliothèque est destinée à la communauté de traitement du langage naturel (NLP) et de recherche d'informations (IR) qui a beaucoup de potentiel et de flexibilité pour la modélisation, l'évaluation et le réglage des thèmeatiques.

3.2.3 Google Collab

Google Colab est un service cloud basé sur Jupyter Notebook pour l'apprentissage automatique. Cette plateforme permet l'entrainment de modèles d'apprentissage automatique directement dans le cloud avec $12,72$ Go de RAM, Intel(R) Xeon(R) CPU @ 2.20GHz, $107,77$ Go de disque et GPU P4 de Nvidia.

3.3 Préparation des données

Dans la présente section, nous commençons par la présentation de l'ensemble de données que nous avons choisi pour conduire notre étude expérimentale. Ensuite nous entamons la phase de prétraitement de cet ensemble de données qui est une tâche très intéressante dans tout projet d'analyse de données. La qualité du prétraitement influence directement les résultats escomptés.

3.3.1 Ensemble des donnée

L'ensemble de données que nous utilisons est une collection d'articles antérieurs de la conférence NIPS (maintenant connue sous le nom de conférence NeurIPS). Le Professeur Sam Royce a compilé une série de documents de conférence NIPS. Cet ensemble de données est téléchargeable à partir du lien suivant : `https://cs.nyu.edu/~roweis/data.html`

Ces données textuelles ont été extraites à l'aide de la technique de reconnaissance optique de caractères (Optical Character Recognition, OCR). Le recours à cette technique a entraîné des mots mal orthographiés, des lettres manquantes, ...

L'ensemble de données comprend 1747 articles et 78908 termes. Chaque article est placé dans son propre fichier texte. La procédure de téléchargement et de lecture des données article par article est montrée par le Listing 3.1.

Listing 3.1 – Télechargemant et lecteur des données.

```
code:

import os
import numpy as np
import pandas as pd
```

```
from google.colab import drive
drive.mount('/content/drive') DATA_PATH = '/content/drive/My

Drive/nipstxt/'
print(os.listdir(DATA_PATH))

['nips01', 'nips04', 'MATLAB_NOTES', 'nips10','nips02', 'idx','nips11
','nips03', 'nips07', 'README_yann', 'nips05','nips12', 'nips06','RAW
_DATA_NOTES', 'orig', 'nips00', 'nips08', 'nips09']
 folders = ["nips{0:02}".format(i) for i in range(0,13)]

 # (b) Lire tous les textes dans la liste.

papers = []
for folder in folders:
 file_names = os.listdir(DATA_PATH + folder)
 for file_name in file_names:
  with open(DATA_PATH + folder + '/' + file_name, encoding='utf8',

  errors='ignore', mode='r+') as f:
    data = f.read()
  papers.append(data)
# (c) Nombre de documents lus.
print('nombre de doc : ' , len(papers))

Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
['README_yann', 'MATLAB_NOTES', 'RAW_DATA_NOTES', 'nips05', 'nips09',
'orig', 'nips08', 'nips04', 'nips11', 'nips07', 'nips06', 'nips10',
'nips12', 'nips00', 'idx', 'nips01', 'nips02', 'nips03']

nombre de doc : 1747
```

Selon les recherche dans la modélisation thématique, un total de 1747 articles est très suffisant pour mener notre étude. À titre d'illustration, le Listing 3.2 présente un fragment de text u d'un article de recherche de notre ensemble de données.

Listing 3.2 – Présentation du contenu d'un article.

```
# (d) Afficher le contenu d'un doc
print('le contenu d"un doc : ')
```

```
print(papers[0][:100

le contenu d"un␣doc␣:
1
CONNECTIVITY␣VERSUS␣ENTROPY
Yaser␣S.␣Abu-Mostafa
California␣Institute␣of␣Technology
Pasadena,␣CA␣91125
ABSTRACT
How␣does␣the␣connectivity␣of␣a␣neural␣network␣(number␣of␣synapses␣per
neuron)␣relate␣to␣the␣complexity␣of␣the␣problems␣it␣can␣handle

(measured
by
the␣entropy)?␣Switching␣theory␣would␣suggest␣no␣relation␣at␣all,since

all␣Boolean
functions␣can␣be␣implemented␣using␣acircuit␣withvery␣low␣connectivity

(e.g.,
using␣two-input␣NAND␣gates).␣However,␣for␣a␣network␣that␣learns␣a

problem
from␣examples␣using␣locallearning␣rule,␣we␣prove␣that␣the␣entropy␣of

the
problem␣becomes␣a␣lower␣bound␣for␣the␣connectivity␣of␣the␣network.
INTRODUCTION

The␣most␣distinguishing␣feature␣of␣neuralnetworks␣is␣their␣ability␣to

spontaneously␣learn␣the␣desired␣function␣from␣'training'samples,␣i.e.

their
ability
to␣program␣themselves.␣Clearly,␣a␣given␣neural␣network␣cannot␣just

learn
any
```

```
function , there must be some restrictions on which networks can learn
which

functions. One obv
```

3.3.2 Prétraitement

Comme nous l'avons déjà indiqué auparavant, la phase de prétraitement est très importante et affecte les résultats de l'expérimentation d'une manière substantielle. Pour cette raison, avant de se lancer dans les expérimentations, nous procédons à un certain traitement préliminaire des textes et quelques changements dans le corpus.

1- Dans une première étape, nous procédons à la tokenisation, lemmatisation et suppression des mot vides ainsi que n'importe quel mot composé d'un seul caractère. Nous utilisons pour cela la bibliothèque nltk (Listing 3.3).

Listing 3.3 – Tokenisation ; lemmatisation et suppression des mots vides.

```
%%time
import nltk
nltk.download('wordnet')
nltk.download('stopwords')
stop_words = nltk.corpus.stopwords.words('english')
wtk = nltk.tokenize.RegexpTokenizer(r'\w+')
wnl = nltk.stem.wordnet.WordNetLemmatizer()
def normalize_corpus(papers):
 norm_papers = []
 for paper in papers:
  paper = paper.lower()
  paper_tokens = [token.strip() for token in wtk.tokenize(paper)]
  paper_tokens = [wnl.lemmatize(token) for token in paper_tokens if
  not token.isnumeric()]
  paper_tokens = [token for token in paper_tokens if len(token) > 1]
  paper_tokens = [token for token in paper_tokens if token not in
  stop_words]
  paper_tokens = list(filter(None, paper_tokens))
  if paper_tokens:
   norm_papers.append(paper_tokens)
 return norm_papers
norm_papers = normalize_corpus(papers)
print(len(norm_papers))
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!

1747

CPU times: user 28.9 s, sys: 229 ms, total: 29.1 s
Wall time: 29.1 s
```

Suite à ces opérations, les articles de recherche deviennent un vecteur de termes, et non pas un text brut. Le Listing 3.4 montre un article de recherche sous forme d'un vecteur de termes.

Listing 3.4 – Exemple d'un article de recherche traité.

```
# affichage d'un papier
print()
print('les mots racines d"un papier id[U+FFFD]: ')
print(norm_papers[0][:50])

 les termes d"un papier :

['connectivity', 'versus', 'entropy', 'yaser', 'abu', 'mostafa',
'california', 'institute', 'technology', 'pasadena', 'ca', 'abstract
',

'doe', 'connectivity', 'neural', 'network', 'number', 'synapsis',
'per', 'neuron', 'relate', 'complexity', 'problem', 'handle',
'measured', 'entropy', 'switching', 'theory', 'would', 'suggest',
'relation', 'since', 'boolean', 'function', 'implemented', 'using',
'circuit', 'low', 'connectivity', 'using', 'two', 'input', 'nand',
'gate', 'however', 'network', 'learns', 'problem', 'example', 'using

']
```

2- Dans cette phase nous essayons d'extraire et de générer des mots et des bi-grammes sous forme de phrases pour chaque article de recherche tokenisé. Nous utilisons pour cela la classe *gensim.models.Phrase*. Le Listing 3.5 montre un exemple de papier où le modèle de génération de phrase est appliqué. À titre d'exemple la phrase "neural_nettwork" sera considéré plutard comme un seul terme.

Listing 3.5 – Construction d'un modèle de phrase bi-gramme.

```
import gensim
bigram = gensim.models.Phrases(norm_papers, min_count=20, threshold=20,
delimiter=b'_')
# higher threshold fewer phrases.
bigram_model = gensim.models.phrases.Phraser(bigram)
# exemple de demonstration
print(bigram_model[norm_papers[0]][:50])
print('Total Vocabulary Size:', len(dictionary))

['connectivity', 'versus', 'entropy', 'yaser', 'abu_mostafa',
'california_institute', 'technology_pasadena', 'ca_abstract', 'doe',
'connectivity', 'neural_network', 'number', 'synapsis', 'per',
'neuron', 'relate', 'complexity', 'problem', 'handle', 'measured',
'entropy', 'switching', 'theory', 'would', 'suggest', 'relation',
'since', 'boolean_function', 'implemented', 'using', 'circuit',
'low', 'connectivity', 'using', 'two', 'input', 'nand', 'gate',
'however', 'network', 'learns', 'problem', 'example', 'using',
'local', 'learning', 'rule', 'prove', 'entropy', 'problem']
Total Vocabulary Size: 78908
```

4- Nous attribuons des numéros uniques à chaque terme/phrase car l'apprentissage automatique ne fonctionne que sur les tenseurs numériques (Listing 3.6).

Listing 3.6 – Numérisation des termes.

```
Sample word to number mappings: (0,'8a'), (1,'abandon'), (2,'able'),
(3, 'abo'), (4, 'abstract'), (5, 'accommodate'), (6, 'accuracy'),
(7,'achieved'),(8,'acknowledgment_thank'),(9,'across'),(10,'active'),
(11, 'activity'), (12, 'actual'), (13, 'adjusted'), (14, 'adjusting')
```

5- Cette phase s'intéresse à l'élagage du corpus. Nous supprimons tous les termes qui apparaissent moins de 20 fois dans tous les documents et tous les termes qui apparaissent plus de 60% dans tous les documents (Listing 3.7). En effet, les seuils choisis sont les plus utilisés dans la littérature.

Listing 3.7 – Filtrage du vocabulaire.

```
# Filtrage des mots qui apparaissent dans moins de 20 fois , ou plus
de 60% des documents.
dictionary.filter_extremes(no_below=20, no_above=0.6)
print('Total Vocabulary Size:', len(dictionary))
```

```
Total Vocabulary Size: 7768
```

6- Cette étape se focalise sur la transformation de corpus en sac de mots. Ceci peut être exploité en utilisant le modèle bag of words tel que montrée dans le Listing 3.8.

Listing 3.8 – Transformation du corpus en sac de mots.

```
# Transformation de corpus en sac de  mots (bag of words
vectors)
bow_corpus = [dictionary.doc2bow(text) for text in norm_corpus_bigr
ams] print(bow_corpus[1][:50])

print([(dictionary[idx] , freq) for idx, freq in bow_corpus[1][:50]])

[(3, 2), (4, 1), (11, 1), (13, 1), (15, 1), (21, 1), (27, 1),(30, 1),
(33, 1), (35, 2), (36, 1), (45, 1), (46, 1), (47, 1),(48, 2),(57, 2),
(64, 7), (67, 4), (71, 3), (78, 1), (92, 3), (94, 2),(97, 9),(104,1),
(113, 1), (115, 4), (118, 4), (120, 2), (124, 4), (125, 3), (128, 1),
(133, 5), (136, 3), (138, 1), (142, 3), (144, 6), (152, 2), (155, 1),
(156, 1), (158, 1), (162, 1), (168, 1), (172, 1),(177, 12),(199, 1),
(203, 14), (204, 1), (213, 1), (215, 1), (218, 4)]
[('ability', 2), ('abu_mostafa', 1), ('afosr', 1), ('air_force', 1),
('american_institute', 1), ('appendix', 1), ('assume', 1),
('asymptotic', 1), ('axe', 1), ('become', 2),('becomes', 1),('bt',1),
('ca_abstract', 1), ('california_institute', 1), ('cannot', 2),
('complete', 2), ('connected', 7), ('consider', 4),
('corresponding',3), ('denote', 1), ('ea', 3), ('ed', 2), ('element',
9), ('environment', 1), ('expected', 1), ('expression', 4), ('fact',
4), ('final', 2), ('fixed', 4), ('follows', 3), ('furthermore', 1),
('get', 5), ('go', 3), ('going', 1), ('hand', 3), ('hence', 6),
('implemented', 2), ('independent', 1), ('independently', 1),
('interested', 1), ('ity', 1), ('know', 1), ('le', 1), ('let', 12),
('need', 1), ('neuron', 14), ('next_section', 1), ('occur', 1),
('office_scientific', 1), ('otherwise', 4)]
```

Après toutes ces étapes, les articles de recherche peuvent être représentées sous forme de sac de mots. Ainsi, nous pouvons commencer à appliquer les méthodes de modélisation thématique LSA et LDA sujettes de notre étude comparative.

3.4 Expérimentation

Comme mentionné précédemment, le but de cette étude expérimentale consiste à comparer les deux modèles, LSA et LDA. Pour ce faire, nous utilisons les mesures de performances *temps d'exécution* et *cohérence thématique* [Mohammed and Al-augby, 2020].

Cohérence thématique

La cohérence thématique est une mesure généralement utilisée pour évaluer les modèles d'analyse thématique en mesurant le degré de similarité sémantique des mots dans une thèmatique. Il existe deux mesures de cohérence thématique (mesure intrinsèque (UMass), mesure extrinsèque (UCI)). Les deux mesures calculent la somme des scores par paires sur les mots de deux manières différentes. La valeur élevée du modèle de score de cohérence thématique représente un bon modèle thématique.

- UCI : Dans la mesure UCI, chaque terme unique est associé à chaque autre terme. il utilise des informations mutuelles ponctuelles : $UCI = log\frac{P(w_i,w_j)+1}{P(w_i)+P(w_j)}$.
  Où :
  — $P(w)$ représente la probabilité que w soit présent dans un document aléatoire.
  — $P(w_i, w_j)$ représente la probabilité que w_i et w_j soient présents dans le même document.
- Umass : Elle sert à comparer seuls les termes précédents et suivants d'un terme donné : $Umass = log\frac{D(w_i,w_j)+1}{D(w_i)}$.
  Où :
  — $D(w_i)$ représente le nombre de documents contenant le terme w_i.
  — $D(w_i, w_j)$ représente le nombre de documents contenant les termes w_i et w_j.

Si le score UMass est bas et le score UCl en cohérence est élevé, meilleur est le modèle.

3.4.1 Évaluation LSA

Nous commençons par la méthode LSA (Voir Section 2.2.1).

À titre d'illustration, nous commençons par la mise en oeuvre de LSA en utilisant Gensim et en extrayant les thèmatiques à partir des articles de recherche de NIPS. Nous prenons, à titre d'essai, le nombre de thèmes égale à 10 (Listing 3.9).

Listing 3.9 – Programme LSA.

```
# Latent Semantic Analysis

%%time
TOTAL_TOPICS = 10
lsi_bow = gensim.models.LsiModel(bow_corpus, id2word=dictionary,
num_topics=TOTAL_TOPICS, onepass=True, chunksize=1740,
```

```
power_iters=1000)
for topic_id, topic in lsi_bow.print_topics(num_topics=10, num_words=
20):
 print('Topic #'+str(topic_id+1)+':')
 print(topic)
 print()
```

Après l'exécution du modèle LSA nous affichons les thématiques et les termes associés (Listing 3.10).

Listing 3.10 – Affichage des thématiques et les termes associés (LSA).

```
Topic #1:
0.214*"unit" + 0.212*"state" + 0.187*"training" + 0.178*"neuron" +

0.162*"pattern" + 0.144*"image" + 0.140*"vector" + 0.125*"feature" +

0.122*"cell" + 0.110*"layer" + 0.101*"task" + 0.097*"class" +

0.091*"probability" + 0.089*"signal"+ 0.087*"step"+ 0.086*"response"

+ 0.085*"representation" + 0.083*"noise" + 0.082*"rule" +

0.081*"distribution"

Topic #2:
-0.489*"neuron" + -0.396*"cell" + 0.255*"state" + -0.191*"response" +

0.187*"training" + -0.171*"stimulus" + -0.118*"activity" +

0.110*"class" + -0.099*"spike" + -0.097*"pattern" + -0.096*"circuit"+

-0.096*"synaptic" + 0.096*"vector" + -0.090*"firing"+ -0.090*"signal"

+ -0.088*"visual" + 0.084*"classifier" + 0.082*"action"+ 0.080*"word"

+ -0.078*"cortical"
.
.
.
Topic #10:
```

```
0.525*"word" + -0.262*"training" + 0.249*"vector" + -0.226*"task" +

-0.190*"pattern" + -0.172*"classifier" + 0.142*"recognition" +

0.138*"sequence" + -0.133*"control" + 0.130*"node" + 0.129*"cell" +

0.116*"circuit"+ -0.112*"action"+ 0.110*"character" + -0.109*"rule" +

-0.101*"neuron" + 0.095*"matrix" + 0.088*"hmm" + 0.085*"structure" +

0.083*"chip"
```

Nous remarquons dans ces résultats que chaque thématique est un mélange de termes liés à des poids (positifs et négatifs). D'après les recherches actuelles, plus il y a de poids du terme est grand, plus le terme est important dans la thématique et les termes associés ont le même signe. Pour que la thématique principale ait deux sous-thématiques différentes basées sur le signe. Afin de clarifier cela davantage, nous regroupons les signes négatifs et les signes positifs (Listing 3.11, Listing 3.12).

Listing 3.11 – Programme de regroupement des signes des termes (positifs/négatifs).

```
for n in range(TOTAL_TOPICS):
 print('Topic #'+str(n+1)+':')
 print('='*50)
 d1 = []
 d2 = []
 for term, wt in lsi_bow.show_topic(n, topn=20):
  if wt >= 0:
    d1.append((term, round(wt, 3)))
  else:
    d2.append((term, round(wt, 3)))
 print('Direction 1:', d1)
 print('-'*50)
 print('Direction 2:', d2)
 print('-'*50)
 print()
```

Listing 3.12 – Résultat de regroupement des signes.

```
Topic #1:
==================================================
```

```
Direction 1: [('unit', 0.214), ('state', 0.212), ('training', 0.187),
('neuron', 0.178), ('pattern', 0.162), ('image', 0.144), ('vector',
0.14), ('feature', 0.125), ('cell', 0.122), ('layer', 0.11), ('task',
0.101), ('class', 0.097), ('probability', 0.091), ('signal', 0.089),
('step', 0.087), ('response', 0.086), ('representation', 0.085),
('noise', 0.083), ('rule', 0.082), ('distribution', 0.081)]
--------------------------------------------------------------------
Direction 2: []
--------------------------------------------------------------------

Topic #2:
====================================================================

Direction 1: [('state', 0.255), ('training', 0.187), ('class', 0.11),
('vector', 0.096), ('classifier', 0.084), ('action', 0.082), ('word',
0.08)]
--------------------------------------------------------------------
Direction 2: [('neuron', -0.489), ('cell', -0.396), ('response',
-0.191), ('stimulus', -0.171),('activity', -0.118),('spike', -0.099),
('pattern', -0.097), ('circuit', -0.096), ('synaptic', -0.096),
('firing', -0.09), ('signal', -0.09), ('visual', -0.088),('cortical',
-0.078)]
--------------------------------------------------------------------
.
.
.
--------------------------------------------------------------------
```

```
Topic #10:
================================================================
Direction 1: [('word', 0.525), ('vector', 0.249), ('recognition',
0.142), ('sequence', 0.138), ('node', 0.13), ('cell', 0.129),
('circuit', 0.116), ('character', 0.11), ('matrix', 0.095), ('hmm',
0.088), ('structure', 0.085), ('chip', 0.083)]
----------------------------------------------------------------
Direction 2: [('training', -0.262), ('task', -0.226), ('pattern',
-0.19), ('classifier', -0.172), ('control', -0.133), ('action',
-0.112), ('rule', -0.109), ('neuron', -0.101)]
----------------------------------------------------------------
```

Maintenant, les résultats sont plus clairs. Nous appliquons la méthode SVD pour obtenir les trois matrices principales (U, S et V^T) (Listing 3.13).

Listing 3.13 – Application de la méthode SVD.

```
term_topic = lsi_bow.projection.u
singular_values = lsi_bow.projection.s
topic_document = (gensim.matutils.corpus2dense(lsi_bow[bow_corpus],
len(singular_values)).T / singular_values).T
term_topic.shape, singular_values.shape, topic_document.shape
((7756, 10), (10,), (10, 1740))
```

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
0	0.038	0.002	0.009	-0.077	-0.010	-0.018	0.030	-0.072	0.000	0.005
1	0.022	0.000	-0.022	0.008	0.011	-0.016	0.013	-0.017	0.001	0.007
2	0.021	-0.028	-0.004	0.003	-0.003	0.000	-0.008	0.009	0.008	-0.014
3	0.024	-0.048	0.011	0.003	-0.023	0.044	-0.002	-0.011	-0.003	0.016
4	0.032	-0.020	-0.013	0.013	-0.003	0.058	-0.006	-0.050	-0.061	0.036
5	0.085	-0.183	-0.003	-0.022	-0.019	0.307	-0.087	-0.137	-0.040	0.020

FIGURE 3.1 – Matrice (U) document-thématique du modèle LSA.

Nous pouvons maintenant découvrir les thématiques les plus importantes pour certains articles (13, 250 et 500) (Listing 3.14).

Listing 3.14 – Exemple d'application modèle LSA

```
document_numbers = [13, 250, 500]
for document_number in document_numbers:
 top_topics = list(document_topics.columns[np.argsort(-
    np.absolute(
  document_topics.iloc[document_number].values))[:3]])
 print('Document #'+str(document_number)+':')
 print('Dominant Topics (top 3):', top_topics)
 print('Paper Summary:')
 print(papers[document_number][:500])
 print()
```

```
Document #13:
Dominant Topics (top 3): ['T7', 'T1', 'T5']


Document #250:
Dominant Topics (top 3): ['T10', 'T4', 'T8']

Document #500:
Dominant Topics (top 3): ['T8', 'T10', 'T1']
```

En observant la répartition des termes dans chacune des thèmatique identifiées dans la sortie précédente, nous pouvons conclure qu'elle est parfaitement logique.

- Les thèmatiques 7, 1 et 5 sont dominées dans l'article $n°13$, qui parle largement des neurones, des cellules, du cortex, des stimuli, ... (aspects liés aux réseaux de neurones).
- Les thèmatiques 10, 4 et 8 ont été dominés dans l'article $n°250$, qui traite de la reconnaissance d'objets, de la classification d'images et de la visualisation avec les réseaux de neurones. Cela correspond au thèmatique de l'article sur la reconnaissance d'objets.
- L'article $n°500$ est dominé par les thèmatiques 1, 8 et 10, qui traitent des signaux, de la tension, des puces, des circuits, ... Ceci est conforme au thèmatique de l'article sur les réglages des paramètres pour les circuits analogiques.

Nous arrivons maintenant à l'évaluation proprement dite de la méthode LSA. Nous exécutons le modèle en changeant à chaque fois le nombre de thèmatiques $(10, 15, 20$ et $25)$. À chaque appel nous mesurons le temps d'exécution et la cohérence thématique (UCI et Umass). les résultats de l'expérimentation sont affichés dans le Listing 3.15.

Listing 3.15 – Mesures de performance de modèle LSA.

```
LSA

TOTAL TOPICS 10
Temps  412.42017889022827
Avg. Coherence Score (UCl): 0.35925037431863055
Avg. Coherence Score (UMass): -1.1104557035454037


TOTAL TOPICS 15
Temps  436.1610405445099
Avg. Coherence Score (UCl): 0.336322122430547
Avg. Coherence Score (UMass): -1.1412153252197128


TOTAL TOPICS 20
Temps  472.64821338653564
Avg. Coherence Score (UCl): 0.30440461736431523
Avg. Coherence Score (UMass): -1.2373167344248612


TOTAL TOPICS 25
Temps  497.8480808734894
Avg. Coherence Score (UCl): 0.28775407518470075
Avg. Coherence Score (UMass): -1.2514845749100685
```

Nous remarquons à travers les résultats que :

- Il existe une relation directe entre le nombre de thématique et le temps d'exécution.
- Le score UCl diminue lorsque le nombre de thèmatiques augmente, à cause des dimensions élevées des matrices manipulées par LSA.
- La meilleure performance pour LSA selon la cohérence thématique UCI a été lorsque le nombre de thématiques était de 10 et la valeur atteinte est égale à (0.355290).
- Le score $UMass$ est en baisse constante.

Nous concluons que le modèle LSA n'est pas bon quand les dimensions des matrices sont élevées, notamment quand le nombre de thème augmente.

3.4.2 Évaluation LDA

De la même manière que lors de l'évaluation de LSA, nous procédons pour LDA. Nous commençons par une illustration de l'exécution du modèle LDA sur les mêmes données et pour

un nombre de thèmatiques de 10 (Listing 3.16).

Listing 3.16 – Affichage des thématiques et les termes associés (LDA).

```
# Latent Dirichlet Allocation
print("LDA ")
list_time_lda = []
list_cv_lda = []
list_U_lda = []
list_topic_nb = []
for TOTAL_TOPICS in [10]:
 start_time = time.time()
 list_topic_nb.append(TOTAL_TOPICS)
 lda_model = gensim.models.LdaModel(corpus=bow_corpus,id2word=diction
ary,
 chunksize=1747, alpha='auto',
 eta='auto', random_state=42,
 iterations=500, num_topics=TOTAL_TOPICS,
 passes=20, eval_every=None)
 # LDA Topics with Weights
 topics_with_wts = [item[0] for item in topics_coherences]
 print('LDA Topics with Weights')
 print('='*50)
 for idx, topic in enumerate(topics_with_wts):
 print('Topic #'+str(idx+1)+':')
 print([(term, round(wt, 3)) for wt, term in topic])
 print()


LDA Topics with Weights
==================================================
Topic #1:
[('let', 0.006), ('vector', 0.006), ('training', 0.006), ('bound',
0.005), ('class', 0.005), ('convergence', 0.005), ('probability',
0.005), ('theorem', 0.005), ('linear', 0.005), ('optimal', 0.005),
('approximation', 0.004), ('rate', 0.004), ('size', 0.004),
('distribution', 0.004), ('node', 0.003), ('consider', 0.003),
('defined', 0.003), ('theory', 0.003), ('unit', 0.003), ('sample',
0.003)]
Topic #2:
```

```
[('training', 0.014), ('task', 0.008), ('rule', 0.008), ('feature',
0.006), ('trained', 0.005), ('test', 0.005), ('training_set', 0.004),
('experiment', 0.004), ('table', 0.004), ('classification', 0.004),
('unit', 0.004), ('target', 0.004), ('expert', 0.004), ('prediction',
0.004), ('pattern', 0.004), ('vector', 0.003), ('class', 0.003),
('generalization', 0.003), ('representation', 0.003), ('hidden_unit',
0.003)]
.
.
.
Topic #10:
[('circuit', 0.016), ('chip', 0.014), ('analog', 0.01), ('current',
0.009), ('voltage', 0.009), ('bit', 0.007), ('signal', 0.006),
('neuron', 0.006), ('noise', 0.005), ('node', 0.005),
('implementation', 0.004), ('threshold', 0.004), ('design', 0.004),
('computation', 0.004), ('gate', 0.004), ('state', 0.004),
('transistor', 0.004), ('vector', 0.004), ('digital', 0.004),
('synapse', 0.004)]
```

Nous remarquons que les résultats sont plus clairs que les résultats de LSA. Cela correspond également au principe de base de la LDA : Chaque documents textuelle est un mélange de thématiques et chaque thématique est un mélange de termes.

Afin d'évaluer le LDA, nous exécutons le modèle en augmentant à chaque fois le nombre de thématiques (10, 15, 20 et 25) et nous mesurons le temps d'exécution et les scores de cohérence thématique (Listing 3.17).

Listing 3.17 – Mesures de performance du modèle LDA.

```
LDA


TOTAL TOPICS 10
Temps 117.86668968200684
Avg. Coherence Score (UCI): 0.4823930870317753
Avg. Coherence Score (UMass): -1.0129342034425377



TOTAL TOPICS 15
Temps 167.12815380096436
Avg. Coherence Score (UCI): 0.4841282849252695
Avg. Coherence Score (UMass): -1.0967049989078734
```

```
TOTAL TOPICS 20
Temps  187.51737570762634
Avg. Coherence Score (UCI): 0.4883688338927379
Avg. Coherence Score (UMass): -1.2590425994435521


TOTAL TOPICS 25
Temps 198.32835865020752
Avg. Coherence Score (UCI): 0.4644865864933877
Avg. Coherence Score (UMass): -1.25357471756478
```

Nous remarquons à travers les résultats obtenus que :

- Il existe une relation directe entre le nombre de thèmes et le temps d'exécution.
- Le score UCl s'améliore avec l'augmentation du nombre de thèmes, puis diminue lorsque le nombre de thèmes arrive à 25.
- Le score $UMass$ diminue avec le nombre de thèmes, et augmente à nouveau lorsque le nombre de thèmes atteint 25.
- La meilleure performance pour LDA selon la cohérence UCI a été lorsque le nombre de thèmes était de 20 et la valeur atteinte (0.488368)

On conclut de ces observations que le modèle LDA n'est pas affecté par une augmentation du nombre de thèmes.

3.4.3 Comparaison de LDA et LSA

Nous comparons les modèles LSA et LDA en fonction des mesures de performance pour voir quel modèle est le meilleur.

1. Temps d'exécution (par seconde) (Table 3.1)

TABLE 3.1 – Comparaison du temps d'exécution entre LSA et LDA.

Nombre de thèmatiques	LSA	LDA
10	412.42	117.87
15	436.16	167.13
20	472.65	187.52
25	497.85	198.33

Du point de vue temps d'exécution, il est claire que le modèle LDA est le meilleur quel que soit le nombre de thèmes.

2. Mesure UCI (Table 3.2)

TABLE 3.2 – Comparaison de LSA et LDA selon le score UCL.

Nombre de thèmatiques	LSA	LDA
10	0.36	0.48
15	0.34	0.48
20	0.30	0.49
25	0.29	0.46

Le score UCI est élevé dans le modèle LDA et en constante amélioration, contrairement au score UCI dans le modèle LSA, qui empirait à chaque augmentation du nombre de thèmatiques.

3. Mesure $UMass$ (Table 3.3)

TABLE 3.3 – Comparaison de LSA et LDA selon le score UMass.

Nombre de thématiques	LSA	LDA
10	-1.11	-1.01
15	-1.14	-1.09
20	-1.24	-1.26
25	-1.25	-1.25

Pour le nombre de thématiques 10-15 : le modèle LSA a excellé dans le score $UMass$.
Pour le nombre de thématiques 20-25 : Convergence du score $UMass$ avec une légère supériorité du modèle LSA.

3.5 Conclusion

Dans ce chapitre nous avons conduit une étude expérimentale comparative de deux méthodes modélisation thématique sur un ensemble d'articles scientifiques de la conférence Nips.

Cette étude a commencé par un certain nombre de prétraitements de l'ensemble de données pour le préparer à la phase d'apprentissage de deux modèles (LSA, LDA). Les résultats en terme de temps d'exécution et de cohérence thématique ont été évalués à l'aide des valeurs de cohésion et du temps d'exécution révèlent que :

- L'ensemble de données joue un rôle important dans la qualité de l'analyse des thèmatiques pour les deux techniques et il est également nécessaire d'augmenter la taille de l'ensemble de données pour améliorer les performances.

- Pour les deux techniques utilisées, l'étape de prétraitement est une étape essentielle car elle permet une bonne réduction de dimensionnalité et supprime les mots inutiles des données textuelles non structurées.
- La mesure de la cohérence des thématiques peut être considérée comme un moyen utile de comparer différentes techniques de modélisation de thématiques en fonction de leur interopérabilité humaine, ce qui conduit à fournir une vue claire et donc à prendre de bonnes décisions.
- LDA a donné de meilleurs résultats que LSA en se basant sur le score UCI sur un ensemble de données des articles scientifiques de la conférence NIPS.

Conclusion générale

La modélisation thématique est utilisée pour découvrir des structures sémantiques latentes qui existent dans une collection de données.

Dans ce mémoire nous avons introduit le concept de modélisation thématique, ses différentes approches et ses domaines d'applications.

En tant que contribution, nous avons conduit une étude expérimentale comparative entre la méthode d'analyse sémantique latente (LSA) et celle d'allocation de Dirichlet latente (LDA) sur un corpus des articles scientifiques de la conférence NIPS (Neural Information Processing Systems).

Les résultat obtenus en terme de temps d'exécution et de cohérence thématique sont en faveur de la méthode LDA.

A titre de perspectives et dans le cadre de la promotion de la langue arabe, nous envisageons travailler sur des corpus en Arabe.

Nous recommandons aussi de travailler avec des corpus de taille importante pour obtenir de meilleurs performances.

Une autre perspective consiste à étudier profondément la modélisation thématique en utilisant l'apprentissage profond.

Bibliographie

[Blei, 2012] Blei, D. M. (2012). Probabilistic topic models. *Commun. ACM*, 55(4) :77–84.

[Blei et al., 2003] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3(null) :993–1022.

[Brandt, 1976] Brandt, S. (1976). *Statistical and computational methods in data analysis ; 2nd ed.* North-Holland, Amsterdam.

[Deerwester et al., 1990] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 41(6) :391–407.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1) :1–38.

[Dou et al., 2007] Dou, Z., Song, R., and Wen, J.-R. (2007). A large-scale evaluation and analysis of personalized search strategies. In *WWW '07 : Proceedings of the 16th international conference on World Wide Web*, pages 581–590. ACM Press. Note that Zhicheng Dou's email address has been changed to zhichdou at microsoft.com now. The old email douzc@yahoo.com.cn is unavailable. Sorry for this.

[Durrett, 2019] Durrett, R. (2019). *Probability : Theory and Examples.* Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 5 edition.

[Erlin, 2017] Erlin, M. (2017). Topic Modeling, Epistemology, and the English and German Novel.

[Griffiths and Steyvers, 2004] Griffiths, L. T. and Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, pages 5228–5235.

[Hofmann, 1999] Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, page 50–57, New York, NY, USA. Association for Computing Machinery.

[Hofmann, 2001] Hofmann, T. (2001). Unsupervised learning by probabilistic latent semantic analysis. *Mach. Learn.*, 42(1–2) :177–196.

[Jidigam et al., 2015] Jidigam, R., Austin, T., and Stamp, M. (2015). Singular value decomposition and metamorphic detection. *Journal of Computer Virology and Hacking Techniques*, 11 :203–216.

[Joachims, 1998] Joachims, T. (1998). Text categorization with suport vector machines : Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML '98, pages 137–142, London, UK, UK. Springer-Verlag.

[Lee and Seung, 1999] Lee, D. D. and Seung, H. S. (1999). Learning the parts of objects by nonnegative matrix factorization. *Nature*, 401 :788–791.

[Lu et al., 2011] Lu, Y., Mei, Q., and Zhai, C. (2011). Investigating task performance of probabilistic topic models : An empirical study of plsa and lda. *Information Retrieval*, 14(2) :178–203.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. In Bengio, Y. and LeCun, Y., editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

[Miller, 2013] Miller, I. (2013). Rebellion, crime and violence in qing china, 1722–1911 : A topic modeling approach. *Poetics*, 41 :626–649.

[Mohammed and Al-augby, 2020] Mohammed, S. and Al-augby, S. (2020). Lsa lda topic modeling classification : Comparison study on e-books. pages 2502–4752.

[Moody, 2016] Moody, C. E. (2016). Mixing dirichlet topic models and word embeddings to make lda2vec.

[Newman and Block, 2006a] Newman, D. J. and Block, S. (2006a). Probabilistic topic decomposition of an eighteenth-century american newspaper. *J. Assoc. Inf. Sci. Technol.*, 57(6) :753–767.

[Newman and Block, 2006b] Newman, D. J. and Block, S. (2006b). Probabilistic topic decomposition of an eighteenth-century american newspaper. *J. Assoc. Inf. Sci. Technol.*, 57(6) :753–767.

[Park and Ramamohanarao, 2009] Park, L. A. F. and Ramamohanarao, K. (2009). The sensitivity of latent dirichlet allocation for information retrieval. In Buntine, W., Grobelnik, M., Mladenić, D., and Shawe-Taylor, J., editors, *Machine Learning and Knowledge Discovery in Databases*, pages 176–188, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Peter and Kp, 2009] Peter, R. and Kp, S. (2009). Evaluation of svd and nmf methods for latent semantic analysis. *SHORT PAPER International Journal of Recent Trends in Engineering*, 1.

[Salton et al., 1975] Salton, G., Wong, A., and Yang, C. S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11) :613–620.

[Song et al., 2010] Song, W., Zhang, Y., Liu, T., and Li, S. (2010). Bridging topic modeling and personalized search. *International Conference on Computational Linguistics*, pages 1167–1175.

[Strang, 2006] Strang, G. (2006). *Linear algebra and its applications*. Thomson, Brooks/Cole, Belmont, CA.

[Talley et al., 2011] Talley, E. M., Newman, D., Mimno, D., Herr, B. W., Wallach, H. M., Burns, G. A. P. C., Leenders, A. G. M., and McCallum, A. (2011). Database of nih grants using machine-learned categories and graphical clustering. *Nat Meth*, 8(6) :443–444.

[Vosecky et al., 2014] Vosecky, J., Leung, K. W.-T., and Ng, W. (2014). Collaborative personalized Twitter search with topic-language models. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 53–62. ACM.

[Yang et al., 2011] Yang, T.-I., Torget, A., and Mihalcea, R. (2011). Topic modeling on historical newspapers. In *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 96–104, Portland, OR, USA. Association for Computational Linguistics.

[Yi and Allan, 2009] Yi, X. and Allan, J. (2009). A comparative study of utilizing topic models for information retrieval. *ECIR*, pages 29–41.

[Zeng et al., 2012] Zeng, Q. T., Redd, D., Rindflesch, T. C., and Nebeker, J. R. (2012). Synonym, topic model and predicate-based query expansion for retrieving clinical documents. In *AMIA 2012, American Medical Informatics Association Annual Symposium, Chicago, Illinois, USA, November 3-7, 2012*. AMIA.

[Zhai and Massung, 2016] Zhai, C. and Massung, S. (2016). *Text Data Management and Analysis : A Practical Introduction to Information Retrieval and Text Mining*, volume 16 of *ACM Books*. Morgan & Claypool, San Rafael, CA.