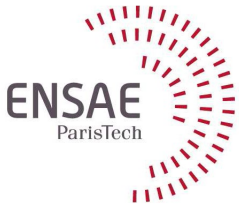
ENSAE ParisTech

Projet de Statistique Appliquée

Application de l'algorithme d'allocation de Dirichlet latente



Mariya Benjelloun
Benjamin Muller
Thomas Perret
Léa Vitrac

Tuteurs :
MM. Vincent Cottet
et James Ridgway
Mai 2015

Remerciements

Nous tenons à adresser nos sincères remerciements à ceux qui ont contribué à l'élaboration de ce projet, tout particulièrement Monsieur Vincent Cottet et Monsieur James Ridgway pour le temps qu'ils ont su nous accorder.

Nous remercions également nos professeurs de Statistiques qui ont su nous donner les outils nécessaires à la compréhension d'un tel sujet, en particulier Monsieur Nicolas Chopin ainsi que Monsieur Alexandre Tsybakov.

Enfin, nous remercions le président du jury qui nous fait l'honneur d'examiner notre projet.

Sommaire

Introduction

Ces trente dernières années, et en particulier les années 2000, ont été marquées par un phénomène nouveau ; le développement du web. Le nombre d'internautes a très fortement augmenté depuis 2001. Internet est aujourd'hui utilisé dans le monde entier, par plus de la moitié de la population. La multiplication des utilisateurs s'est bien entendu accompagnée de l'explosion du contenu disponible sur la toile.

Dans ce contexte, une question s'est très rapidement posée : comment trouver une information pertinente pour l'internaute au milieu de ces milliards d'informations accessibles ? C'est le problème que résolvent par exemple les moteurs de recherche. Lorsque l'on entre un mot-clé dans la barre de recherche, Google nous renvoie la liste des sites qu'il juge les plus pertinents en rapport avec notre saisie, en seulement quelques centièmes de secondes. Quels algorithmes sont cachés derrière une simple recherche sur Internet ?

Au développement exponentiel du volume d'informations accessible à tous, une réponse a été apportée : le ***machine learning***. Cette discipline est définie par Arthur Samuel, l'un des pionniers du domaine, comme *"un domaine d'étude qui permet aux ordinateurs d'apprendre sans être explicitement programmés"* [1]. On peut compléter cette définition en disant que ce que l'ordinateur apprend, il l'apprend à partir de *données*. En d'autres termes, le *machine learning* est un domaine de l'informatique regroupant des algorithmes permettant à un ordinateur d'analyser des *données d'apprentissage*, d'en déduire des règles (par exemple sur la structure de ces données), et finalement d'appliquer celles-ci à de nouvelles données.

Deux branches principales se distinguent en *machine learning* : les algorithmes dits d'***apprentissage supervisé*** dans lesquels la variable à expliquer est préalablement définie et les algorithmes d'***apprentissage non supervisé*** dans lesquels il n'existe pas de telle variable. Au sein de chacune de ces branches, on peut vouloir faire de la *prédiction* ou de la *classification (ou clustering)*.

Les algorithmes de prédiction permettront par exemple de prédire le prix d'un appartement à Paris en fonction de sa surface, de sa localisation, de son état de propreté... L'*output*, ici le prix d'un appartement à Paris, est bien défini, et connu. Les prédictions sont souvent effectuées à l'aide de régressions.
Les algorithmes de classifications, eux, doivent être capables de classer des objets en différentes catégories, selon leurs caractéristiques, sans qu'on ait préalablement défini ces catégories. Si par exemple on travaille sur un échantillon d'arbres, on va donner différentes informations à l'ordinateur, comme la hauteur de l'arbre, la circonférence de son tronc, sa couleur, la taille de ses feuilles, leur forme, leur couleur... A l'issu de l'algorithme, l'ordinateur aura regroupé les arbres ayant des caractéristiques semblables, et si on lui donne les caractéristiques d'un nouvel arbre, il pourra déterminer à quelle espèce celui-ci appartient. L'*output*, ici l'espèce de l'arbre, n'est pas défini au préalable dans cette catégorie d'algorithmes. Cela illustre ainsi l'apprentissage non supervisé.

L'algorithme d'***allocation de Dirichlet latente*** (ou LDA pour *latent Dirichlet allocation*), que nous avons étudié dans le cadre de ce projet, entre dans la deuxième catégorie, les algorithmes d'apprentissage non supervisé. Appliqué à des *données textuelles*, il va nous permettre de mettre en évidence le ou les thèmes principaux abordés dans un document à partir des mots contenus dans celui-ci. Il va également pouvoir classer chaque mots au sein de thèmes, *qu'il déterminera lui-même*.

On voit bien ici l'application directe qu'un tel algorithme peut avoir dans le cadre d'une simple recherche de document par mots-clés. Cependant, cet algorithme a de nombreux autres domaines d'application, comme l'analyse d'image ou encore l'analyse vocale, que nous ne développerons pas ici.

1. *"A field of study that gives computers the ability to learn without being explicitly programmed."*

1 Contexte

Études préalables

Le modèle d'allocation de Dirichlet latente (*LDA* a été présenté pour la première fois en 2003 par David Blei, Andrew Ng et Michael Jordan dans un article du *Journal of Machine Learning Research* et a depuis inspiré de nombreuses publications qui ont repris de près ou de loin leurs recherches.

Les travaux de Blei, Ng et Jordan se sont basés sur de nombreux articles de recherche écrits auparavant. Ils se sont notamment inspirés du travail réalisé par Baeza-Yates et Ribeiro-Neto en 1999 sur la *récupération de l'information*, et de celui réalisé par Salton et McGill en 1983 sur la *représentation d'un corpus*. Selon cette méthodologie, chaque document du corpus est réduit à un vecteur dont chaque case correspond à un mot du dictionnaire et contient un nombre représentant la fréquence d'apparition de ce mot.

Cependant, cette modélisation d'un corpus a ses limites. D'une part, elle n'est pas très avantageuse en termes de représentation du corpus ; elle ne propose pas de générer un document selon une loi par exemple. On connaît seulement les probabilités de chaque mot au sein d'un document. D'autre part, elle permet difficilement de comparer des documents *entre eux*. Pour pallier ces limites, d'autres études ont été faites dans le domaine de la modélisation de corpus. On pense notamment au travail de Deerwester et al. en 1990, connu sous le nom de *latent semantic indexing (ou LSI)*. Mais l'une des avancées les plus significatives dans ce domaine a été celle proposée par Hofmann en 1999. Celui-ci a présenté le modèle *probabilistic LSI (ou pLSI)*, comme une alternative au modèle *LSI*. Cette approche modélise chaque mot d'un document comme étant un échantillon d'un modèle de mélange où les composantes du mélange peuvent être interprétées comme des variables aléatoires multinomiales représentant les *topics*. Ainsi, on fait l'hypothèse que chaque mot est généré à partir d'un unique *topic*, et on retrouve pour un même document des mots appartenant à des *topics* différents. ***Chaque document est donc réduit à une loi de distribution sur un ensemble fixé de topics.***

Ces différents travaux ont donc été un point de départ pour Blei, Ng et Jordan qui ont pu développer le modèle, aujourd'hui connu sous le nom d'***allocation de Dirichlet latente***.

Avancées et limites du modèle

L'originalité du travail de Blei, Ng et Jordan a été d'utiliser des méthodes de statistiques bayésiennes. Il s'agit donc de comprendre l'intérêt de ces méthodes par rapport aux approches classiques et les éventuels inconvénients qu'elles présentent.

La méthode bayésienne est tout d'abord un moyen naturel de combiner l'information du *prior* avec les données, le tout dans un cadre théorique très précis. En effet, on peut intégrer des informations passées d'un paramètre et former une distribution *a posteriori* pour une analyse future. De plus, lorsque de nouvelles observations sont disponibles, cette distribution *a posteriori* peut être réutilisée comme un *prior*.

Par ailleurs, les méthodes bayésiennes étant des méthodes d'apprentissage *non supervisé*, le risque de sur-apprentissage est nul.

Enfin, les réponses fournies sont interprétables assez facilement. En effet, l'algorithme renvoie deux hyper-paramètres :

— α nous permet de connaître la probabilité d'apparition de chaque *topic* au sein d'un document. On peut par exemple savoir si le document abordera plutôt le thème du travail, de la politique ou de la médecine, ou si il abordera tout autant les trois sujets.

— β nous donne la probabilité d'apparition de n'importe quel mot au sein de n'importe quel *topic*. Ainsi, en considérant les 5 ou 10 mots les plus représentés dans un *topic*, on reconnaît facilement le *topic* en question.

Il est important de noter qu'un grand avantage du modèle *LDA* est qu'il permet à chaque document d'aborder ***plusieurs topics***, contrairement à la majorité des modèles précédents.

Le modèle de l'allocation de Dirichlet latente a été facilement étendu pour obtenir des modèles plus complexes. On peut se baser sur lui pour modéliser les relations entre *topics*, ou pour trouver de façon générative le nombre de *topics* d'un corpus de textes par exemple.

Néanmoins, l'approche bayésienne présente quelques inconvénients. Tout d'abord le peu d'information au sujet du *prior* rend son choix difficile et la moindre erreur peut affecter fortement les lois *a posteriori*. De plus, une telle approche va souvent de pair avec un fort coût en temps de calcul, spécifiquement dans les modèles avec de nombreux paramètres. Enfin, se pose aussi le problème des hyper-paramètres qui interviennent dans les calculs et requièrent une bonne intuition au préalable afin que les résultats obtenus coïncident avec le modèle préétabli.

2 Principe de l'inférence variationnelle

Un des grands enjeux de notre étude, va être de calculer des grandeurs associés à des distributions de probabilités L'inférence variationnelle regroupe un ensemble de méthodes et techniques permettant d'*approcher* des intégrales qui ne sont pas calculables de façon exacte. Avec le développement du *machine learning*, ces techniques connaissent un grand essor, en particulier dans les modèles statistiques complexes. Elles sont souvent utilisées dans deux situations :

— pour approcher numériquement une distribution *a posteriori* de variables latentes ;

— pour dériver une borne inférieure de la vraisemblance des données observées.

Nous nous plaçons ici dans un modèle bayésien. Notre objectif va donc être, une fois notre modèle établi (ce que nous ferons dans la Partie 4), d'***approcher la distribution* a posteriori *des variables latentes et des paramètres de notre modèle sachant les observations***.

2.1 Un problème d'inférence variationnelle

Plus concrètement, considérons le modèle bayésien suivant.

Les N variables observées seront appelées $X = \{X_1, X_2, \cdots, X_N\}$. On les suppose indépendantes et identiquement distribuées. Les variables latentes seront notées $Z = \{Z_1, Z_2, \cdots, Z_N\}$ et les paramètres seront regroupés dans θ. Notons Y la concaténation des variables latentes et des paramètres, soit $Y = \{Z|\theta\}$. Dans notre problème de données textuelles, les variables observées seront par exemple les mots ou les documents, les variables latentes seront les *topics* associés à chaque mot ou document.

Le modèle étant bayésien, nous allons poser une loi *a priori* pour la loi jointe de (X, Y), que nous noterons $p(X, Y)$. Notre objectif est de déterminer la loi *a posteriori* $p(Y|X)$, pour pouvoir connaître la répartition des *topics* au sein de notre corpus de textes. Pour cela, nous allons chercher à maximiser la vraisemblance de $(Y|X)$.

Par définition, approcher la distribution $p(Y|X)$ par une distribution $q(Y)$ sur un ensemble de distributions $\mathcal{E}$ revient à minimiser la distance de Kullback-Leibler entre $p(Y|X)$ et l'ensemble $\mathcal{E}$.

$$\begin{aligned}\arg\max_{q\in\mathcal{E}} E_q\left[\log\left(\frac{p(Y|X)}{q(Y)}\right)\right] &= \arg\min_{q\in\mathcal{E}} E_q\left[\log\left(\frac{q(Y)}{p(Y|X)}\right)\right] \\ &= \arg\min_{q\in\mathcal{E}} KL[q(Y)||p(Y|X)].\end{aligned}$$

La ***distance de Kullback-Leibler*** (aussi appelée *divergence de Kullback-Leibler*), est une mesure de dissimilarité entre deux distributions de probabilités. Elle est appelée distance par abus de langage, mais ce n'en est pas une ; elle n'est pas symétrique et ne respecte pas l'inégalité triangulaire.

Plus formellement, si P et Q sont deux mesures, et si Q est absolument continue par rapport à P, alors :

$$KL[P||Q] = -\int log\frac{dP}{dQ}dP.$$

Sinon,

$$KL[P||Q] = \infty.$$

Par ailleurs, pour toute distribution $q(Y)$, nous pouvons décomposer la log-loi marginale de X comme suit :

$$\begin{aligned}\ln(p(X)) &= E_q\left[\ln(p(X))\right] \\ &= E_q\left[\ln\left(\frac{p(X,Y)}{p(Y|X)}\right)\right] \\ &= E_q\left[\ln\left(\frac{p(X,Y)}{q(Y)}\right)\right] - E_q\left[\ln\left(\frac{p(Y|X)}{q(Y)}\right)\right] \\ &= E_q\left[\ln\left(\frac{p(X,Y)}{q(Y)}\right)\right] + KL[q||p(Y|X)].\end{aligned}$$

Dans l'égalité ci-dessus, $\ln p(X)$ est constante. De plus, on reconnaît dans le terme de droite la distance de Kullback-Leiber que nous cherchons minimiser. Il est donc équivalent de minimiser cette distance ou de maximiser le premier terme de la somme ci-dessus. Et maximiser cette grandeur sur l'ensemble $\mathcal{E}$ revient à approximer la loi p(X).
Nous cherchons donc à estimer la quantité donnée par la formule suivante :

$$\hat{q} = \arg\max_{q\in\mathcal{E}} E_q\left[\ln\left(\frac{p(X,Y)}{q(Y)}\right)\right].$$

2.2 Restriction de l'ensemble des distributions

En pratique, calculer de manière exacte cette grandeur est souvent hors de portée. Nous allons donc l'approcher en restreignant l'ensemble des distributions considérées.

Malheureusement, dans le cas général et en particulier dans notre application à des données textuelles, nous ne savons pas calculer ce maximum de vraisemblance si l'ensemble E est quelconque. Nous allons donc l'approcher en restreignant l'ensemble des distributions considérées.

L'ensemble des distributions $\mathcal{E}$ que nous avons choisi est la famille des lois ***indépendantes*** (ou factorisables), c'est-à-dire les distributions q qui s'écrivent :

$$q(y) = q(\theta, z_1, \cdots, z_N) = q_\theta(\theta)\cdot\prod_{i=1}^{N} q_i(z_i).$$

Grâce à cette restriction et à l'expression de $\hat{q}$, on peut facilement isoler les lois marginales $q_i(z_i)$ et $q_\theta(\theta)$ pour tout i dans $[\![1;N]\!]$:

$$\begin{aligned}q_\theta(\theta) &\propto e^{E_Z[ln(p(X,\theta,Z))]}, \\ q_{Z_i}(z_i) &\propto e^{E_{\theta,Z_{j(j\neq i)}}[ln(p(X,\theta,Z))]}.\end{aligned}$$

Ceci n'est pas une solution explicite du modèle, car le calcul de chaque distribution de probabilité des variables latentes ou du paramètre nécessite la connaissance préalable des distributions des autres variables latentes et paramètre. Pour résoudre ce problème et approximer ces distributions, nous allons donc mettre en place un algorithme itératif. Nous commencerons par initialiser toutes les distributions, puis à les recalculer une par une en fonction des autres distributions. La convergence est garantie par les travaux de Boyd et Vandenberghe de 2004.

Calculer explicitement les lois marginales de θ et des $(Z_i)_{i\in[\![1;N]\!]}$ nécessite la mise en place d'un modèle, c'est-à-dire d'hypothèses sur les lois suivies par nos données textuelles. C'est ici que le modèle LDA intervient.

3 Exemple simplifié : le mélange de gaussiennes

Avant d'étudier le modèle LDA dans toute sa complexité, nous avons dans un premier temps appliqué et implémenté un algorithme d'inférence variationnelle sur des données correspondant à un mélange de gaussiennes. L'objectif de cette démarche était de mettre en oeuvre et de tester l'algorithme sur une base de données plus simple. Nous souhaitions également bien comprendre quelles étaient les variables observées, les variables latentes, et comment calculer les différents paramètre du modèle dans un cadre simplifié.

3.1 Présentation du modèle

Reprenons les notations de la section précédente. Supposons que les $(X_i)_{i\in[\![1;N]\!]}$ soient i.i.d, selon le mélange gaussien $p \times \mathcal{N}(0,1) + (1-p) \times \mathcal{N}(\theta, 1)$. p est un réel de $]0;1[$ connu, et θ un paramètre à estimer. Cela signifie que chaque X_i a la probabilité p d'être tiré selon la loi $\mathcal{N}(0,1)$, et la probabilité $(1-p)$ d'être tiré selon la loi $\mathcal{N}(\theta, 1)$.

Variables latentes

Considérons la variable latente $(Z_i)_{i\in[\![1;N]\!]}$ définie comme suit :

$$Z_i = \begin{cases} 1 & \text{si } X_i \text{ a été tiré selon } \mathcal{N}(0,1), \\ 0 & \text{sinon.} \end{cases}$$

$(Z_i)_{i\in[\![1;N]\!]}$ est donc une variable latente du modèle. L'évaluer permettrait de déterminer selon quelle loi ont été tirés les X_i (ie. à quel groupe ils appartiennent, ou dans notre exemple de données textuelles, à quel *topic* ces mots appartiennent).

Comme nous nous sommes placés dans un modèle bayésien, nous devons imposer une loi *a priori* sur le paramètre θ. Nous choisissons de prendre une *loi normale centrée réduite*.

Par la suite, on notera $\phi(X, a, b)$ la valeur de la densité d'une gaussienne d'espérance a et de variance b pour l'observation X.

Loi jointe

La *loi de* $X_i|Z_i, \theta$ s'écrit alors naturellement ainsi :

$$p(X_i|Z_i,\theta) \quad \propto \quad \phi(X_i,0,1)^{1_{Z_i=1}} \cdot \phi(X_i,\theta,1)^{1_{Z_i=0}}.$$

Par ailleurs, grâce à l'hypothèse d'indépendance des distributions de la Partie 2.2, on peut écrire la *loi jointe de* (X, Z, θ) de la façon suivante :

$$\begin{aligned} p(X,Z,\theta) \quad &\propto \quad p(\theta) \cdot \prod_{i=1}^{N} p(X_i|Z_i,\theta) \cdot p(Z_i), \\ p(X,Z,\theta) \quad &\propto \quad \phi(\theta,0,1) \cdot \prod_{i=1}^{N} \left[\phi(X_i,0,1)^{1_{Z_i=1}} \cdot \phi(X_i,\theta,1)^{1_{Z_i=0}}\right] \left[p^{1_{Z_i=1}} \cdot (1-p)^{1_{Z_i=0}}\right]. \end{aligned}$$

Résultat sur les lois marginales

Grâce aux calculs détaillés en Annexe 1, on obtient les résultats suivants concernant les distributions marginales du paramètre θ et des variables latentes $(Z_i)_{i\in[\![1;N]\!]}$:

$$\begin{cases} q_\theta \sim \mathcal{N}(m, \sigma^2), \\ q_{Z_i} \sim \mathcal{B}\left(\frac{p\cdot\phi(X_i,0,1)}{p\cdot\phi(X_i,0,1)+\frac{1-p}{\sqrt{(2\pi)}}*e^{-\frac{(X_i-m)^2+\sigma^2}{2}}}\right), \end{cases}$$

avec :

$$\begin{cases} m = \frac{\sum_i(1-p_i)X_i}{1+\sum_i(1-p_i)}, \\ \quad \text{et} \\ \sigma^2 = \frac{1}{1+\sum_i(1-p_i)}. \end{cases}$$

Comme mentionné dans la Partie 2.2, il s'agit à présent d'initialiser q_θ et les $(q_{Z_i})_{i\in[\![1;N]\!]}$, puis de les recalculer successivement à l'aide des autres distributions. Cet algorithme est décrit plus précisément dans la partie suivante.

3.2 Implémentation sous R

On commence par générer les données selon le mélange de gaussiennes décrit dans le paragraphe précédent. La fonction ***GenDonnées*** est décrite en pseudo-code dans l'Algorithme 1. Celle-ci prend en argument le nombre de données souhaitées n, la probabilité p de suivre une loi gaussienne centrée réduite dans le mélange gaussien, et le paramètre θ qui est l'espérance de la deuxième gaussienne du mélange. En sortie, on obtient donc un vecteur de données de taille n.

GenDonnées (n, p, θ)
U $\leftarrow n$ nombres générés selon $\mathcal{B}$(p)
W $\leftarrow n$ nombres générés selon $\mathcal{N}$(0,1)
V $\leftarrow n$ nombres générés selon $\mathcal{N}(\theta$,1)
X $\leftarrow U \cdot W + (1 - U) \cdot V$
Return X

Algorithm 1: Génération des données

Puis on crée une deuxième fonction, ***VarBayes***, décrite dans l'Algorithme 2. Celle-ci prend en argument les données générées par la fonction ***GenDonnees***, la probabilité p de suivre une loi gaussienne centrée réduite dans le mélange gaussien, et un critère d'arrêt *pas*. Le critère d'arrêt que nous avons choisi est la convergence de m ; si la différence absolue des valeurs prises par m dans deux itérations successives de l'algorithme est inférieure au *pas* choisi, alors l'algorithme s'arrête. La terminaison de la fonction est garantie par la convergence de m. La fonction ***VarBayes*** renvoie les valeurs terminales prises par les $(p_i)_{i\in[\![1;N]\!]}$, m et σ^2.

VarBayes (X, p, pas)
initialisation
$n \leftarrow taille(X)$;
$p_i^{(0)} \leftarrow 0.5 \quad \forall i$;
$m^{(0)} \leftarrow 0$;
$\sigma^{2(0)} \leftarrow 1$;
while $|m^{(t-1)} - m^{(t-2)}| < pas$ **do**
  $m^{(t)} = \frac{\sum_{i=1}^{N}(1-p_i)\cdot X_i}{1+\sum_{i=1}^{N}(1-p_i)}$;
  $\sigma^{2(t)} = \frac{1}{1+\sum_{i=1}^{N}(1-p_i)}$;
  for $i \in [1; N]$ **do**
    $p_i^{(t)} = \frac{p\cdot\phi(X_i,0,1)}{p\cdot\phi(X_i,0,1)+\frac{1-prob}{\sqrt{2\pi}}\cdot e^{-\frac{(X_i-m_i^{(t)})^2+\sigma^{2(t)}}{2}}}$;
  end
end
Return $m^{(T)}, \sigma^{2(T)}, (p_i^{(T)})_{i\in[1;N]}$

Algorithm 2: Algorithme du *Variational Bayes*

Nous avons choisi arbitrairement les valeurs d'initialisation suivantes :
- $m = 0$;
- $\sigma^2 = 1$;
- $p_i = 0.5 \, \forall i$.

L'algorithme implémenté sous R est disponible en Annexe 2.

3.3 Résultats

Génération des données

Pour générer les données, on prend :
- $\theta = 3$;
- $n = 1000$;
- $p = 0.2$.

Pour rappel, cela signifie qu'on a généré 1000 données selon le mélange gaussien $0.2 \cdot \mathcal{N}(0, 1) + 0.8 \cdot \mathcal{N}(3, 1)$. La densité des données générées à l'aide de ces valeurs est représentée dans la Figure 1.

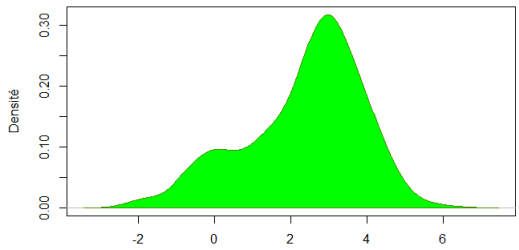


FIGURE 1 – Densité des données générées selon un mélange gaussien.

Convergence de l'algorithme du *variational Bayes*

Puis nous avons appliqué à ces données notre algorithme du *variational Bayes*, dans le but de retrouver θ ainsi que les p_i. Les résultats obtenus avec les valeurs d'initialisations présentées dans le paragraphe précédent, ainsi qu'en prenant un pas (ou critère d'arrêt de l'algorithme du *variational Bayes*) de 0.001 sont présentés ci-dessous.

Tout d'abord, nous avons observé la convergence effective des différentes valeurs recherchées dans l'algorithme. Nous n'avons pas affiché les valeurs initiales, mais seulement les valeurs prises par les différentes variables à partir de la première itération de l'algorithme. L'algorithme converge très vite, et afficher la première valeur rendrait les suivantes illisibles.

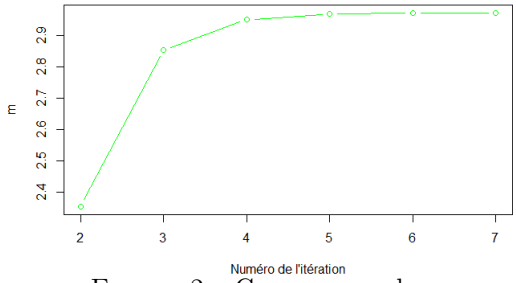


FIGURE 2 – Convergence de m.

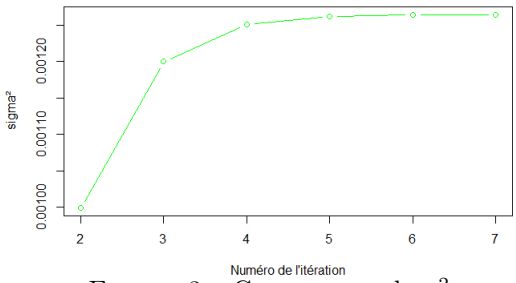


FIGURE 3 – Convergence de σ^2.

On constate bien une convergence très rapide des variables vers leur valeur finale. En seulement 6 itérations de l'algorithme, les valeurs finales sont quasiment atteintes. Les résultats sont tout à fait cohérents avec la théorie, puisqu'on obtient les résultats suivants, présentés dans les Graphiques 2 et 3 :

— $m = 2.97$ (la valeur recherchée étant 3) ;
— $\sigma^2 = 0.001$.

La loi *a posteriori* de θ est donc quasiment un Dirac en 3, ce qui est bien la valeur recherchée.

On remarque également une convergence rapide des p_i. Le premier exemple présenté dans la Figure 4 est la probabilité p_1 associée à l'observation $X_1 = 4.26$. Finalement, on obtient une probabilité de $6.5 \cdot 10^{-5}$ que X_1 suive la première loi normale centrée réduite du mélange, c'est-à-dire une probabilité presque égale à 1 qu'elle suive la seconde gaussienne, centrée en 3. Le second exemple présenté dans la Figure 5 est la probabilité p_2 associée à l'observation $X_2 = -1.96$. Finalement, on obtient une probabilité de 1 (à 10^{-4} près) que X_2 ait été générée selon la loi normale centrée réduite. Les résultats sont cohérents avec l'intuition.

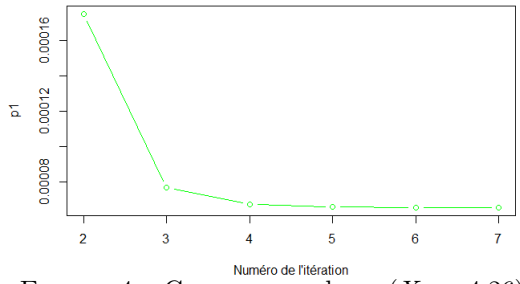


FIGURE 4 – Convergence de p_1 ($X_1 = 4.26$).

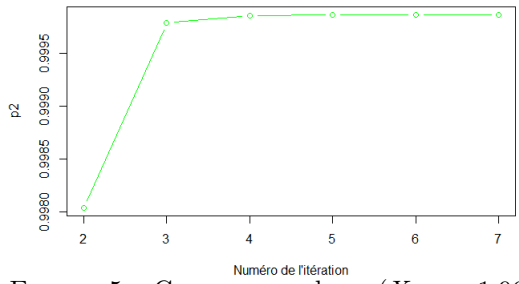


FIGURE 5 – Convergence de p_2 ($X_2 = -1.96$).

Distribution des p_i

La Figure 6 présente les résultats obtenus sur la répartition des $(p_i)_{i\in[1;N]}$. Les observations dont la probabilité p_i obtenue à l'issue de l'algorithme est supérieure à 0.5 sont représentés en rouge, et celles dont elle est inférieure à 0.5 sont représentés en bleu. On remarque que les observations supérieures à 0.9 environ sont plutôt attribuées à la loi $\mathcal{N}(3,1)$, alors que les autres sont plutôt attribuées à la loi $\mathcal{N}(0,1)$. L'algorithme est bien cohérent avec l'intuition, puisque plus l'observation X_i est élevée, moins il est probable que celle-ci ait suivi la première gaussienne du mélange, centrée en 0. Si cette courbe est à peu près centrée autour de 0.9 et pas autour du milieu de 0 et 3, c'est parce que l'algorithme tient compte du fait que la probabilité de suivre la première gaussienne est seulement de 0.2, alors que la probabilité de suivre la seconde est de 0.8.

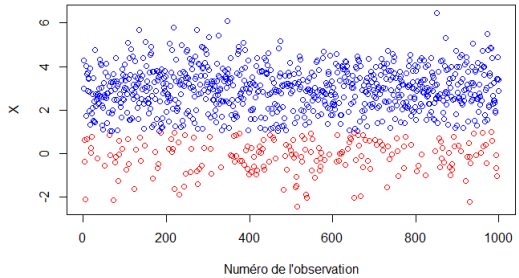


FIGURE 6 – Représentation des X_i selon qu'ils sont probablement tirés selon la gaussienne centrée réduite (en rouge) ou la deuxième (en bleu).

Comparaison des p_i trouvés et de la distribution selon laquelle ont été tirés les X_i.

Il s'agit maintenant de comparer les p_i obtenus avec l'algorithme avec la réalité. Les Figures 7 et 8 montrent la répartition des p_i pour les observations ayant été tirées selon une loi $\mathcal{N}(0,1)$ et $\mathcal{N}(3,1)$ respectivement.

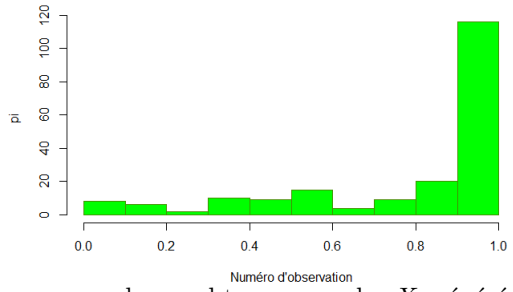


FIGURE 7 – Histogramme des p_i obtenus pour les X_i générés selon la loi $\mathcal{N}(0,1)$

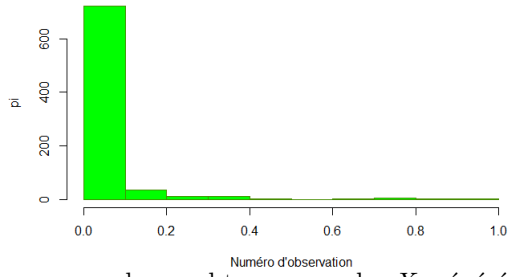


FIGURE 8 – Histogramme des p_i obtenus pour les X_i générés selon la loi $\mathcal{N}(3,1)$

Les résultats obtenus sont satisfaisants. Parmi les 801 observations générées selon la loi $\mathcal{N}(3,1)$, 785 ont une probabilité p_i inférieure à 0.5 à l'issue de l'algorithme, ce qui représente 2% d'erreur. De même, parmi les 199 observations générées selon la loi $\mathcal{N}(0,1)$, 164 ont une probabilité p_i supérieure à 0.5 à l'issue de l'algorithme, ce qui représente 17% d'erreur.

Par ailleurs, on remarque que cet algorithme est un véritable outil de décision. En effet, la majorité des p_i sont très proches de 0 ou de 1, ce qui nous assure clairement qu'une observation est tirée selon l'une ou l'autre des gaussiennes du mélange.

Sensibilité de l'algorithme aux valeurs initiales

Finalement, nous avons modifié les différentes valeurs initiales $m^{(0)}$, ${\sigma^2}^{(0)}$ et $(p_i^{(0)})_{i\in[\![1;N]\!]}$. Nous avons fait varier $m^{(0)}$ et ${\sigma^2}^{(0)}$ de -1000 à $+1000$, et les $(p_i^{(0)})_{i\in[\![1;N]\!]}$ de 0 à 1. Les résultats sont clairs : la moyenne obtenue pour θ à l'issue de tous ces algorithmes est ***la même à*** 10^{-6} ***près***. L'algorithme est donc robuste aux valeurs d'initialisation choisies arbitrairement.

Lien avec le modèle LDA

Ici, nos observations étaient tirées selon des lois gaussiennes. Dans le modèle LDA, les mots seront tirés selon des lois multinomiales. De même, le paramètre θ suivait une loi *a priori* $\mathcal{N}(0,1)$, alors que les paramètres θ à l'aide desquels les multinomiales seront définies suivront dans notre modèle des loi de Dirichlet, comme décrit au paragraphe suivant.

4 Modèle d'allocation de Dirichlet latente appliqué à des données textuelles

Le modèle *d'allocation de Dirichlet latente* est un *algorithme génératif probabiliste*, c'est-à-dire qu'il fait des hypothèses sur les distributions de probabilité selon lesquelles ont été générées les données du problème. La seule restriction sur les données observées est qu'elles doivent appartenir à un ensemble fini, ce qui est effectivement le cas dans le cadre d'une analyse de données textuelles.

Avant d'en décrire le fonctionnement précis, commençons par s'en approcher de façon intuitive. Les données vont être considéré comme des "sacs" : cela signifie que nous ne nous considérerons pas les informations contenues dans l'agencement des données les unes par rapport aux autres. Si l'on prend l'exemple de données textuelles, cela revient à considérer des sacs de mots dont la structure linguistique est ignorée. On prend le mot comme une unité d'information *indépendante de sa place dans le document.*

Une collection de mots forme un ***document***. L'ensemble des documents forme un ***corpus***. L'objectif va être d'associer à chacun des mots un ***topic*** - un thème. Cette association apparaîtra sous la forme d'une distribution de probabilité, qui traduira la probabilité d'obtenir notre mot sachant que l'on est dans un certain *topic*, et ce pour chacun des documents. Seule la répartition des *topics* dans tout le corpus sera supposée de loi connu *a priori.*

Une fois le modèle posé, l'objectif de l'apprentissage sera d'approcher la loi *a posteriori* à partir des observations.

4.1 Présentation du modèle

Présentons dans un premier temps les hypothèses et notations nécessaires à la mise en place du modèle LDA.

— Le ***mot*** est modélisé par un nombre compris entre 1 et V. Cela suppose que chaque mot est contenu dans un dictionnaire prédéfini contenant l'ensemble du vocabulaire utilisé dans l'ensemble des documents. Ce nombre sera noté w.

— Un ***document*** sera assimilé à un vecteur de mots noté $d = (w_1, .., w_N)$, où N est le nombre de mots du document.

— Enfin le ***corpus*** correspondra naturellement à un vecteur de nos documents (donc une matrice) noté $D = (d_1, ...d_M)$, où M est le nombre de documents du corpus.

— A chaque mot w_i sera associé un ***topic*** (c'est-à-dire un thème) noté z_i, inconnu *a priori.* Il sera modélisé par un nombre de $[1; K]$. Il faut ici souligner que le nombre de *topics* est supposé fixé dans ce modèle. Cela signifie que nous choisirons *arbitrairement* le nombre de *topics* que l'algorithme devra "apprendre". Ce paramètre devra être testé dans notre implémentation pour trouver celui qui correspond le mieux à nos données. [2]

— La distribution des *topics* au sein d'un document d sera représentée par un vecteur de taille K noté θ_d vérifiant les propriétés naturelles suivantes :

$$\forall i,\ \theta_{i,d} > 0 \text{ et } \sum_i \theta_{i,d} = 1.$$

Cela signifie que la coordonnée $\theta_{i,d}$ représente la proportion du *topic* i dans le document d considéré, sur lequel nous n'avons aucune information au départ.

2. *Les paramètres w (respectivement z) sont des paramètres de multinomiales. Ils seront vus selon leur utilisation, soit comme des nombres compris respectivement entre 1 et V (resp. entre 1 et K), soit comme des vecteurs de taille V (resp. K) dont la seule composante non nulle a pour indice le nombre en question*

— De manière symétrique, nous définirons la variable β qui représente le poids des mots au sein des *topics*. Il s'agit d'une matrice de taille $K \times V$. Les coordonnées $\beta_{i,j}$ correspondent ainsi au poids du mot indicé par i dans le *topic* indicé par j. Là encore, cette distribution est *a priori* inconnue.

Nous allons supposer que le corpus est généré de la façon suivante.

Pour chaque document d du corpus D,

1. Générer $\theta \sim Dir(\alpha)$,
2. Pour chaque mot w_i du document d :
   (a) Générer $z_i \sim Multinomial(\theta)$,
   (b) Générer $w_i \sim Multinomial(\beta_{z_i})$.

Ces hypothèses sont fondamentales pour la compréhension du modèle LDA. Elles correspondent en réalité aux suppositions faites sur la manière dont un texte est créé. Cela revient à supposer qu'une personne qui écrit un texte choisit en premier une distribution de *topics* particulière (dépendant de la distribution préalablement tirée au sort). Ensuite, à chaque fois qu'elle souhaite écrire un mot, cette personne choisit d'abord un *topic*, puis choisit un mot en accord avec le *topic* choisi. Comme nous l'avons mentionné plus haut, aucune information n'est tirée de la position relative des mots les uns par rapport aux autres. Un document est vu comme un "sac de mots". Le LDA est en ce sens un modèle d'analyse textuelle sémantique.

On distingue ainsi une structure "à trois couches" : un mot associé à un *topic* (donné par $z_{n,d}$), un document associé à une distribution de *topics* (donné par θ_d) et enfin le corpus et le dictionnaire décrits par les hyper-paramètres α et β. Ces structures sont reliées par des distribution de probabilités. En ce sens, le modèle LDA est un modèle hiérarchique bayésien dit à trois couches, représenté en Figure 9.

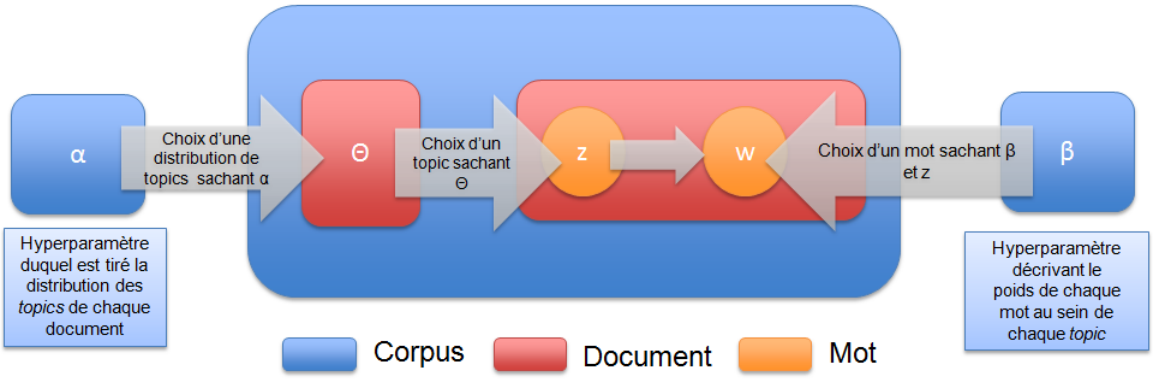


FIGURE 9 – Le LDA, un modèle hiérarchique bayésien à trois couches

Pourquoi avoir choisi ces lois ?

L'objectif du modèle LDA est de mettre en lumière les différents *topics* d'un corpus de documents. Il est ainsi naturel de considérer que nous en choisissons un parmi K, chacun d'entre eux étant pondérés au sein d'un document. Parallèlement, le choix du *topic* étant fait, il semble naturel de considérer que chaque mot du dictionnaire a une certaine probabilité d'être utilisé lorsque celui-ci est abordé, et que cette probabilité dépend seulement du *topic* lui-même et non du document. Dans ces deux cas, l'utilisation d'une distribution multinomiale apparaît

naturelle.

L'utilisation d'une Dirichlet semble moins évidente à première vue. La première raison est d'ordre pratique. En effet, la loi de Dirichlet est conjuguée à la loi multinomiale (cf. encadré ci-dessous). Cela simplifie significativement les calculs, puisque la loi *a posteriori* a la même forme que la loi *a priori*. Cette raison étant donnée, nous pouvons tout de même donner une interprétation du paramètre α en relation avec la structure des données. Lorsque l'on simule un vecteur θ selon cette loi, $\sum_{i=1}^{K} \alpha_i$ mesure l'homogénéité des θ_i. Plus ce nombre est grand, plus les θ_i sont proches les uns des autres. Plus il est petit, plus les θ_i sont majoritairement proches de 0. Ainsi, du point de vue de la structure du corpus, plus $\sum_{i=1}^{K} \alpha_i$ sera grand, ***plus la distribution des topics selon les documents variera, et inversement***.

θ suit une ***loi de Dirichlet*** de paramètre α ($\theta \sim Dir(\alpha)$) si sa densité est de la forme

$$p(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{K} \alpha_i)}{\prod_{i=1}^{K} \Gamma(\alpha_i)} \cdot \theta_1^{\alpha_1 - 1} \cdot \ldots \cdot \theta_K^{\alpha_K - 1},$$

avec $\alpha_i > 0 \, \forall i$.

Propriété : La loi de Dirichlet est ***conjuguée*** à la loi multinomiale, i.e.

Si $z \sim Multinomiale(\theta)$ et $\theta \sim Dir(\alpha)$, alors $\theta|z$ suit une loi de Dirichlet.

4.2 Expression de la loi *a posteriori* des variables latentes et du paramètre

On en déduit, à l'aide de la formule de Bayes, l'expression de la loi jointe propre à chaque document d :

$$p(z, d, \theta|\alpha, \beta) = p(\theta|\alpha, \beta) \cdot p(z, d|\theta, \alpha, \beta).$$

On suppose les distributions des mots et *topics* indépendantes les unes par rapport aux autres (ce qui correspond à l'hypothèse du "*sac de mots*" du LDA). N_d représente la taille du document d.

$$p(d, z, \theta|\alpha, \beta) = p(\theta|\alpha) \cdot \prod_{1}^{N_d} p(z_i, w_i|\theta, \alpha, \beta).$$

On applique la formule de Bayes pour tout i, c'est-à-dire pour chaque mot du document :

$$p(d, z, \theta|\alpha, \beta) = p(\theta|\alpha) \cdot \prod_{1}^{N_d} p(z_i|\theta, \alpha, \beta) \cdot p(w_i|\theta, \alpha, \beta, z_i).$$

Grâce aux différentes dépendances supposées du modèle, on peut simplifier l'expression de la façon suivante :

$$p(d, z, \theta|\alpha, \beta) = p(\theta|\alpha) \cdot \prod_{1}^{N_d} p(z_i|\theta) \cdot p(w_i|\beta_{z_i}).$$

On en déduit la forme de la loi *a posteriori* suivante :

$$p(d,z,\theta|\alpha,\beta) = \frac{\Gamma(\sum_{i=1}^{K}\alpha_i)}{\prod_{i=1}^{K}\Gamma(\alpha_i)} \cdot \prod_{i=1}^{K}\theta_i^{\alpha_i-1} \cdot \prod_{i=1}^{N_d}\left[\left(\prod_{k=1}^{K}\theta_{d,k}^{z_{i,k}} \cdot \frac{z_{i,k}!}{\prod_{k=1}^{K} z_{i,k}!}\right) \cdot \left(\prod_{n=1}^{N_d}\beta_{z_i,w_n}^{w_n} \cdot \frac{w_{n,d}!}{\prod_{n=1}^{N_d} w_{n,d}!}\right)\right].$$

Elle contient l'ensemble de l'information nécessaire à la description voulue de nos données. L'objectif est à présent de la calculer.

4.3 Inférence variationnelle appliquée au modèle LDA

Le calcul direct de cette loi est rendu inenvisageable car elle contient des paramètres latents ou inobservés β, θ et z. Notre but est donc de l'approcher. Pour cela, nous allons utiliser les méthodes d'*inférence variationnelle* mentionnés dans la Partie 2, et nous ramener à un problème d'optimisation classique non convexe. Nous allons faire quelques hypothèses supplémentaires sur le modèle d'approximation, pour nous permettre de découpler les paramètres connus et latents, comme le montre la Figure 10. θ suivra une loi de Dirichlet de paramètre γ, et les *topics* $(z_1, \cdots, z_N)$ des mots seront générés selon une loi multinomiale de paramètres $(\phi_1, \cdots, \phi_N)$. Les paramètres γ et ϕ seront nos *paramètres variationnels*.

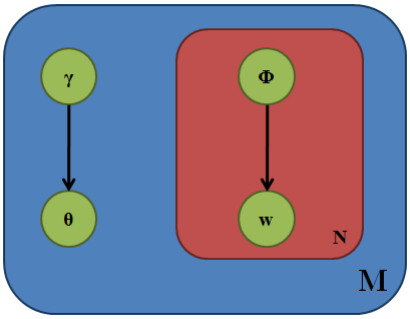


FIGURE 10 – Approximation du modèle LDA [3]

Ainsi, la loi jointe *a posteriori* du paramètre θ et des variables latentes Z_i est donnée par la formule :

$$q(\theta,z|\gamma,\phi) = q(\theta|\gamma) \cdot \prod_{n=1}^{N_d} q(z_n|\phi_n).$$

Dans les calculs suivants, dans le but d'alléger les notations, nous omettrons les paramètres γ et ϕ.

L'inégalité de Jensen nous permet de majorer la quantité suivante :

$$\begin{aligned} log\, p(w|\alpha,\beta) &= log\left(\int_\theta \sum_z p(\theta,z,w|\alpha,\beta)d\theta\right), \\ &= log\left(\int_\theta \sum_z \frac{p(\theta,z,w|\alpha,\beta)\cdot q(\theta,z)}{q(\theta,z)}d\theta\right), \\ &\geq \int_\theta \sum_z q(\theta,z) \cdot log\left(\frac{p(\theta,z,w|\alpha,\beta)}{q(\theta,z)}\right)d\theta. \end{aligned}$$

3. Source : Blei, Ng and Jordan, 2003

Cette dernière inégalité provient de l'inégalité de Jensen, et du fait que :

$$\int_\theta \sum_z q(\theta, z) = 1.$$

On obtient finalement l'inégalité suivante :

$$\begin{aligned} log\, p(w|\alpha, \beta) &\geq E_{q(\theta,z}\left[log\, p(\theta, z, w|\alpha, \beta)\right] - E_{q(\theta,z}\left[log\, q(\theta, z)\right], \\ &= L(\gamma, \phi|\alpha, \beta). \end{aligned}$$

Par ailleurs, on remarque que la différence entre les deux membres de l'inéquation ci-dessus est exactement la distance de Kullback-Leibler entre la distribution variationnelle *a posteriori* des paramètres θ et z et leur vraie distribution *a posteriori*.

$$log\, p(w|\alpha, \beta) - L(\gamma, \phi|\alpha, \beta) = KL\left(q(\theta, z|\gamma, \phi)\,||\,p(\theta, z|w, \alpha, \beta)\right).$$

On remarque dans l'équation ci-dessus que maximiser la vraisemblance $L(\gamma, \phi|\alpha, \beta)$ sur (γ, ϕ) revient à minimiser la divergence de Kullback-Leibler $KL\left(q(\theta, z|\gamma, \phi)\,||\,p(\theta, z|w, \alpha, \beta)\right)$, toujours sur (γ, ϕ). Finalement,

$$\begin{aligned} (\gamma^*, \phi^*) &= \underset{(\gamma,\phi)}{argmin}\, KL\left(q(\theta, z|\gamma, \phi)\,||\,p(\theta, z|w, \alpha, \beta)\right), \\ &= \underset{(\gamma,\phi)}{argmax}\, L(\gamma, \phi|\alpha, \beta). \end{aligned}$$

Vraisemblance du modèle à α et β donnés

Les calculs de la vraisemblance du modèle $L(\gamma, \phi; \alpha, \beta)$ sont présentés dans l'Annexe 3. Le résultat obtenu à l'issue de ces calculs est le suivant (on rappelle que cette vraisemblance est propre à un document qui n'apparaît pas dans les notations pour des raisons de lisibilités) :

$$\begin{aligned} L(\gamma, \phi; \alpha, \beta) = {} & log\,\Gamma\left(\sum_{i=1}^{K}\alpha_i\right) - \sum_{i=1}^{K} log\,\Gamma(\alpha_i) + \sum_{i=1}^{K}(\alpha_i - 1)\cdot\left[\psi(\gamma_i) - \psi\left(\sum_{j=1}^{K}\gamma_j\right)\right] \\ & + \sum_{n=1}^{N}\sum_{i=1}^{K}\phi_{ni}\cdot\left[\psi(\gamma_i) - \psi\left(\sum_{j=1}^{K}\gamma_j\right)\right] \\ & + \sum_{n=1}^{N}\sum_{z_n}\sum_{j=1}^{V} w_{nj}\cdot log\,\beta_{ij}\cdot\phi_{ni} \\ & - log\,\Gamma\left(\sum_{i=1}^{K}\gamma_i\right) + \sum_{i=1}^{K} log\,\Gamma(\gamma_i) - \sum_{i=1}^{K}(\gamma_i - 1)\cdot\left[\psi(\gamma_i) - \psi\left(\sum_{j=1}^{K}\gamma_j\right)\right] \\ & - \sum_{n=1}^{N}\sum_{i=1}^{K}\phi_{ni}\cdot log\,\phi_{ni}. \end{aligned}$$

Maximum de vraisemblance du modèle à α et β donnés

Nous avons d'abord maximisé la vraisemblance en fonction de ϕ_{ni}. Pour rappel :

- ϕ_{ni} est la probabilité que le n-ième mot du document soit généré par le *topic* i.
- ν est l'unique entier tel que $w_n^\nu = 1$. En d'autres termes, w_n est le ν-ième mot du dictionnaire.
- $\beta_{i\nu}$ est donc la probabilité que le mot indexé par ν suive le *topic* j.
- β est une matrice de taille $K \times V$.

— Ψ est la fonction Digamma, qui est la dérivée de la fonction log-Gamma.

Le lagrangien associé s'écrit alors, à une constante près en ϕ_{ni}, où $i \in [\![1; N]\!]$ et $n \in [\![1; V]\!]$:

$$L_{\phi_{ni}} = \phi_{ni} \left(\Psi(\gamma_i) - \Psi \left(\sum_{j=1}^{K} \gamma_j \right) \right) + \phi_{ni} \, log \, \beta_{i\nu} - \phi_{ni} \, log \, \phi_{ni} + \lambda_n \left(\sum_{j=1}^{K} \phi_{ni} - 1 \right).$$

On peut dériver cette expression pour chercher le maximum de vraisemblance, et on obtient la condition du premier ordre suivante :

$$\begin{aligned} \frac{\partial L_{\phi_{ni}}}{\partial \phi_{ni}} &= \Psi(\gamma_i) - \Psi \left(\sum_{j=1}^{K} \gamma_j \right) + log \, \beta_{i\nu} - log \, \phi_{ni} + \lambda_n - 1 \\ &= 0, \end{aligned}$$

d'où

$$\phi_{ni} \propto \beta_{i\nu} e^{\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^{K} \gamma_j\right)}.$$

La formule exacte n'est pas nécessaire car les vecteurs ϕ_n sont normés.

On calcule à présent le lagrangien associé à γ_i, où $i \in [\![1; N]\!]$ s'écrit, à une constante près en γ_i :

$$L_\gamma = \sum_{i=1}^{K} \left(\Psi(\gamma_i) - \Psi \left(\sum_{j=1}^{K} \gamma_j \right) \right) \left(\alpha_i + \sum_{n=1}^{N} \phi_{ni} - \gamma_i \right) - log \, \Gamma \left(\sum_{j=1}^{K} \gamma_j \right) + log \, \Gamma(\gamma_i).$$

Encore une fois, on peut dériver cette expression pour chercher le maximum de vraisemblance, et on obtient la condition du premier ordre suivante :

$$\begin{aligned} \frac{\partial L_\gamma}{\partial \gamma_i} &= \Psi'(\gamma_i) \left(\alpha_i + \sum_{n=1}^{N} \phi_{ni} - \gamma_i \right) - \Psi' \left(\sum_{j=1}^{K} \gamma_j \right) \sum_{j=1}^{K} \left(\alpha_j + \sum_{n=1}^{N} \phi_{nj} - \gamma_j \right), \\ &= 0. \end{aligned}$$

L'une des solutions de ce système à N équations que nous retiendrons dans notre résolution est la suivante :

$$\forall i, \quad \gamma_i = \alpha_i + \sum_{n=1}^{N} \phi_{ni}.$$

Le maximum de vraisemblance du modèle est donc obtenu pour les valeurs de ϕ et γ suivantes :

$$\forall i, \forall n, \quad \phi_{ni} \propto \beta_{i\nu} e^{\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^{K} \gamma_j\right)}, \tag{1}$$

$$\forall i, \quad \gamma_i = \alpha_i + \sum_{n=1}^{N} \phi_{ni}. \tag{2}$$

Maximisation de la vraisemblance du modèle sur α et β

Après avoir maximisé la vraisemblance du modèle à α et β fixés, il faut déterminer ces hyper-paramètres α et β, en maximisant la vraisemblance du modèle par rapport à eux. Pour cela, nous avons fixé γ^* et ϕ^* obtenus par la maximisation de la partie précédente. La vraisemblance du modèle s'écrit de façon claire :

$$L(\alpha, \beta) = \sum_{d=1}^{M} log\, p(w_d|\alpha, \beta).$$

Après calculs, la maximisation par rapport à β donne le résultat suivant :

$$\beta_{ij} \propto \sum_{d=1}^{M} \sum_{n=1}^{N_d} \phi^*_{dni} w^j_{dn}. \tag{3}$$

Le maximum de L en α n'admettant pas d'expression analytique nous avons utilisé une méthode numérique : l'algorithme de Newton-Raphson.

Mise en oeuvre de l'algorithme du LDA

L'algorithme se déroule en deux étapes. Dans un premier temps, il faut fixer des valeurs initiales de α, β, γ et ϕ. Puis, grâce aux équations 1 et 2, on maximise la vraisemblance du modèle à α et β fixés, sur ϕ et γ. Une fois ϕ et γ fixés, on maximise la vraisemblance du modèle sur α et β grâce à l'équation 3 et à la méthode d'optimisation de Newton-Raphson. Puis on réitère la procédure jusqu'à convergence des hyper-paramètres α et β.

Cette procédure est présenté plus précisément dans l'Algorithme 3 ci-dessous.

LDA (*Corpus*)
initialisation
  $\alpha = \alpha^{(0)}$;
  $\beta = \beta^{(0)}$;
while (α, β) *ne converge pas* **do**
  for *chaque document du corpus d* **do**
    while (γ^*_d, ϕ^*_d) *ne converge pas* **do**
      $(\gamma^*_d, \phi^*_d) = \arg\max_{(\gamma,\phi)} L(\gamma, \phi, \alpha^{(t-1)}, \beta^{(t-1)})$
    end
  end
  Return $(\gamma_{tot}, \phi_{tot}) = (\gamma^*_d, \phi^*_d)_d$;
  $(\alpha^{(t)}, \beta^{(t)}) = \arg\max_{(\alpha,\beta)} L(\gamma_{tot}, \phi_{tot}, \alpha, \beta)$
end
Return (α, β)

Algorithm 3: Algorithme du LDA

Les résultats obtenus suite à l'application de cet algorithme à des données simulées d'abord et à des données réelles ensuite sont présentés dans la Partie 5 de ce rapport.

Mise en perspective

En résumé, nous avons implémenté un algorithme selon les principes de l'inférence variationnelle. Notre objectif était d'approcher une distribution décrivant la structure d'un corpus, supposé généré selon les distributions du

modèle LDA.

Observons à présent les pistes possibles d'une extension de notre travail. Elles portent sur deux points précis de notre cheminement. Le premier est le modèle LDA lui-même, dont la portée et la précision peuvent être enrichis. Le second concerne la technique d'approximation de la loi *a posteriori*.

• Extension et souplesse du modèle LDA

On peut citer à présent une amélioration extrêmement riche du modèle LDA. Le LDA, comme nous l'avons vu, aborde le texte d'un point de vue sémantique uniquement. Le texte est vu comme un "sac de mots". Il s'agit ici d'une grande limite de ce modèle, qu'il est toutefois possible de surmonter. C'est ce que font David M. Blei et al. dans leur article *Integrating Topics and Syntax*, en introduisant une variable latente représentant la syntaxe associée à chaque mot et en reliant les différentes valeurs de cette variable par des lois de types chaînes de Markov. Ils différencient ainsi les mots selon leur fonction et leur utilisation dans la phrase, en plus de les différencier selon leur sens. Par exemple, le mot "trained" appartiendra à la fois au champs syntaxique des verbes, et aux champs sémantiques du sport et des statistiques. Ils parviennent ainsi à enrichir le modèle LDA en l'élargissant à une étude syntaxique.

D'autre part, le LDA n'est en aucun cas restreint à des données de type textuel. En l'état, ce modèle est applicable à des données non-textuelles qui respecteraient ce schéma de structure à trois couches, par exemple dans le domaine de l'analyse d'image ou de l'analyse vocale.

De plus, il est toujours possible de rajouter des couches afin de s'adapter au mieux à la structure des données. La limite évidente à cette complexification reste la faisabilité des calculs. C'est ce que font Xiaogang Wang et Eric Grimson dans leur modèle de *Spatial Latent Dirichlet Allocation* qui est une adaptation presque directe du LDA à des données visuelles. En découpant les images en morceaux et en considérant chaque partie comme des documents, ils sont parvenues à mettre en place un algorithme de reconnaissance d'image. Sans entrer dans le détail, cet exemple parmi beaucoup d'autres montre la richesse des modèles hiérarchiques bayésiens.

Enfin et plus généralement, le modèle LDA (ainsi que ses différentes variantes) peut être vu comme un algorithme de réduction dimensionnelle. En effet, ce type de modèle permet une description précise de données extrêmement complexe (des textes, des images, des sons...) à travers quelques paramètres numériques. Dans notre modèle, les données d'entrée sont des mots classés en documents qui, grâce à l'algorithme du LDA, sont décrits par les paramètres α, β, γ, et ϕ.

Pour toutes ces raisons, le LDA apparaît comme une excellente introduction aux modèles hiérarchiques bayésiens plus complexes qui permettent une description toujours plus fine de grands volumes de données.

• Calcul de la loi *a posteriori*

Comme nous avons vu, l'objectif de notre étude réside dans le calcul de la loi *a posteriori* des paramètres et variables latentes. Nous avons choisis d'utiliser l'approximation variationnelle. Il s'agit d'une méthode déterministe d'optimisation, le but étant de maximiser une certaine grandeur sous différentes contraintes. Il est important de noter qu'une autre grande famille de méthodes existe pour approcher les lois recherchées. Il s'agit des méthodes dites de simulation. La méthode de Gibbs-sampling, qui utilise des méthodes de Monte-Carlo par chaînes de Markov, fait partie de cette famille. Si l'on considère une distribution de probabilité donnée p, cette méthode permet de définir une chaîne de Markov dont la distribution stationnaire est p, et donc de simuler une variable aléatoire qui suit p.

Appliqué à notre problème de données textuelles, un tel algorithme présente l'avantage de ne pas avoir à expliciter les paramètres θ et ϕ, mais seulement les variables latentes z_i qui sont simulées successivement par l'algorithme de Gibbs-sampling, comme le montre l'article de Tom Griffiths de 2002. En effet, on parvient à calculer la distri-

bution de probabilité de z_i à i fixé, connaissant w et z_{-i}. Après convergence de l'algorithme, on obtient donc les z_i (ie. les *topics* suivis par les mots w_i).

Le second avantage est que ce type de méthode calcule la loi *a posteriori* exacte au sein d'une famille de lois, et non une approximation comme dans l'inférence variationnelle. Le choix d'une ou l'autre de ces deux grandes familles de méthodes est une question complexe et c'est souvent l'efficacité en temps de calcul qui permet de trancher. Dans le cas de notre modèle LDA, il semble que les deux méthodes soient applicables.

5 Résultats

Interprétation des paramètres variationnels : Le principe de notre algorithme d'approximation variationnelle est d'approcher pour chaque document d $p(z, \theta|\alpha, \beta, d)$ par $q(\theta, z|\gamma(d), \phi(d))$. On peut donc déduire de ϕ et γ des informations sur la structure de chaque document.

• ϕ_n s'interprète comme $p(z_n|w_n, d)$ (car $z \sim Multinomiale(\phi)$), i.e comme la probabilité d'avoir le *topic* z_n sachant le mot considéré, sachant le document considéré.

• γ_n s'interprète de la manière suivante : comme $\theta \sim Dir(\gamma)$ d'où $E(\theta_i) = \frac{\gamma_i}{\sum_i \gamma_i}$.

5.1 Résultats sur les données simulées

Simulation d'un corpus de documents

Dans le but de tester notre algorithme, nous avons commencé par créer nous-même un corpus de textes. Ainsi, nous avons défini nous-mêmes les valeurs des hyper-paramètres α et β, ce qui nous a permis de voir si à l'issue de l'algorithme, nous retrouvions bien les mêmes valeurs. Le code de génération de l'algorithme est disponible en Annexe 4.

Commençons par choisir un dictionnaire très simple, contenant en tout 6 mots (i.e $V = 6$). Ces 6 mots sont regroupés selon deux *topics* (i.e $K = 2$), le bonheur et le travail. Le dictionnaire est divisé comme suit :
- dans le thème "bonheur", nous retrouverons les mots "amour", "bonheur" et "confiance",
- dans le thème "travail", nous retrouverons les mots "chômage", "employeur" et "salaire".

Le dictionnaire est donc un vecteur de taille 6, qui contiendra ces six mots dans l'ordre alphabétique.

Nous avons initialisé β comme suit. Ici, β est une matrice de 2 lignes et 6 colonnes. La première ligne représente le thème du bonheur, tandis que la deuxième représente le thème du travail. La valeur de β_{1j} représente donc la probabilité d'obtenir le mot j lorsque nous nous trouvons dans le thème "bonheur". Les valeurs qui ont été choisies pour β sont les suivantes :
- le mot "amour" a un poids de 0.38 dans le thème du bonheur, et de 0 dans le thème du travail,
- le mot "bonheur" a un poids de 0.57 dans le thème du bonheur, et de 0 dans le thème du travail,
- le mot "chômage" a un poids de 0 dans le thème du bonheur, et de 0.38 dans le thème du travail,
- le mot "confiance" a un poids de 0.05 dans le thème du bonheur, et de 0 dans le thème du travail,
- le mot "employeur" a un poids de 0 dans le thème du bonheur, et de 0.05 dans le thème du travail,
- le mot "salaire" a un poids de 0 dans le thème du bonheur, et de 0.57 dans le thème du travail.

Ces valeurs ont été choisies arbitrairement. Ainsi, dans le thème du travail, le mot qui reviendra le plus souvent sera le mot "salaire", suivi de près par le mot "chômage". Le mot "employeur" n'apparaîtra que très rarement.

Nous avons fixé α à (3,1). Cela signifie que le thème du bonheur sera globalement beaucoup plus abordé que le thème du travail. Nous avons fixé le nombre de documents du corpus à 50 (i.e $M = 50$), puis nous avons généré 50 occurrences de la variable θ selon la loi de Dirichlet de paramètre α. Chaque θ déterminera à quelle fréquence chaque *topic* sera abordé dans chaque document. Par exemple, dans le document 1, les *topics* abordés seront tirés selon une loi multinomiale de paramètre (0.85 , 0.15).

Nous avons ensuite créé une matrice Z de zéros, de 50 lignes et N_{max} colonnes (N_{max} étant le nombre maximal de mots par documents). La valeur de Z_{ij} vaut 1 ou 2, et est le *topic* du mot j dans le document i. Elle a été tirée selon une multinomiale de paramètre θ_i. Puis nous avons généré une matrice $Corpus$, de la même taille que Z, dont chaque ligne représente un document, et chaque case représente un mot représenté par sa place dans le dictionnaire. Ainsi, si Z_{ij} vaut 1, le mot $Corpus_{ij}$ a été généré selon une multinomiale de paramètre la première

ligne de β.

De cette façon, la première ligne de notre corpus (donc le premier document du corpus) une fois traduite selon le dictionnaire est la suivante : "bonheur bonheur salaire chômage salaire amour bonheur salaire bonheur confiance amour bonheur salaire bonheur bonheur bonheur amour bonheur bonheur bonheur amour amour bonheur amour amour bonheur bonheur bonheur bonheur confiance bonheur bonheur amour bonheur bonheur salaire amour bonheur amour bonheur bonheur salaire bonheur bonheur". On remarque par exemple que le mot "employeur" n'apparaît pas : non seulement le thème "travail" a peu de chances d'apparaître dans ce document, mais en plus le mot "employeur" a peu de chances d'apparaître au sein du *topic* "travail".

Le nombre de mots par document a été tiré selon une loi de Poisson de paramètre 40, mais cela n'est pas un paramètre qui doit être déterminé par l'algorithme.

Appication de l'algorithme à ces données simulées

L'algorithme a été implémenté avec un niveau de convergence égal à 0.1.

Après 4 itérations de l'algorithme "Inférence", on retrouve une répartition des *topics* très similaire à celle attendue :

$$\alpha = \begin{pmatrix} 0.947 & 2.987 \end{pmatrix}.$$

Le vecteur β résultant de l'implémentation est décrit dans la Figure 11. Pour rappel, la vraie valeur du vecteur β était :

$$\beta = \begin{pmatrix} 0 & 0 & 0.38 & 0 & 0.05 & 0.57 \\ 0.38 & 0.57 & 0 & 0.05 & 0 & 0 \end{pmatrix},$$

où la première ligne représente le *topic* du bonheur, et la deuxième ligne celui du travail. Les valeurs théoriques et optimisées sont très proches.

	Amour	Bonheur	Chômage	Confiance	Employeur	Salaire
***Topic* 1 : Travail**	0.029	0.011	0.378	0.001	0.037	0.544
***Topic* 2 : Bonheur**	0.302	0.551	0.018	0.056	0.013	0.06

FIGURE 11 – Assimilation des mots du dictionnaire aux *topics* correspondants

On constate ainsi que chaque mot est assigné à son *topic*, ce qui se traduit par une coordonnée en β supérieur pour le *topic* correspondant. On retrouve donc deux *topics* regroupant d'un côté les mots : bonheur, amour, confiance et de l'autre : chômage, salaire, employeur (par ordre d'importance). Avec des données générées selon le modèle LDA, notre algorithme d'inférence variationnelle semble donc donner des résultats très satisfaisants.
Afin de mesurer la convergence de notre algorithme, nous avons calculé la log-vraisemblance $L(\gamma, \phi|\alpha, \beta)$, qui correspond à la grandeur à maximiser dans l'algorithme. Plus précisément, cette quantité est calculée à deux reprises pour chaque itération : après l'inférence variationnelle qui corrrespond à la maximisation de L en ϕ et γ et après l'estimation des paramètres qui correspond à la maximisation de L en α et β. On obtient ainsi des valeurs négatives et croissantes au fil des itérations.
La Figure 12 représente le tracé de la log-vraisemblance à chacune des 4 itérations de l'algorithme d'inférence sur les données simulées.

Nous avons également simulé de nombreux corpus avec des paramètres α et β différents afin de tester au mieux notre algorithme. A chacun de ces tests les résultats étaient satisfaisants (cf Annexe 5 pour un autre exemple).

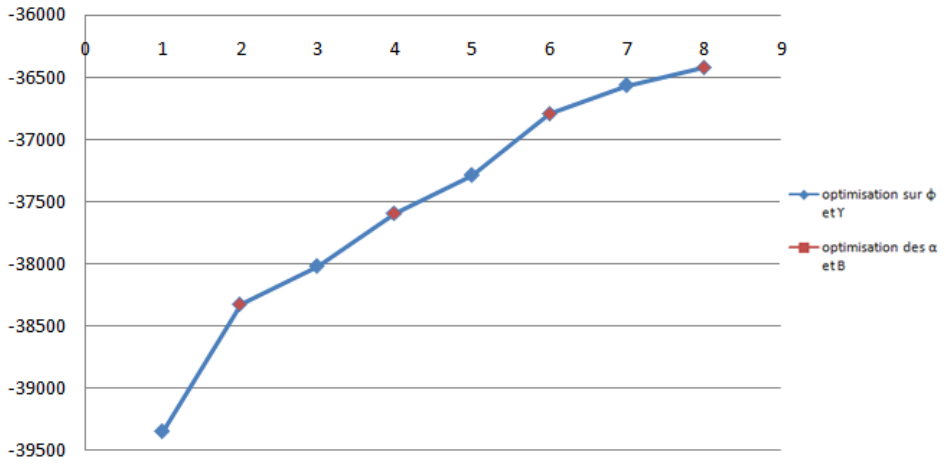


FIGURE 12 – Tracé de la log vraisemblance à chaque étape de l'algorithme

5.2 Résultats sur les données réelles

Présentation des données

Les données utilisées ont été récupérées sur un forum de discussion. Chaque *post* représente un document. Nous avons recensé 200 000 documents de tailles variables en nombre de mots. L'algorithme d'inférence mis en place sur le logiciel R présente une grande complexité ; ainsi, le temps de compilation est très élevé. Nous avons donc décidé de réduire les données à 998 documents. Il a ensuite fallu nettoyer et simplifier cette base de données. Nous avons, dans ce but, supprimé tout d'abord les mots ayant un apport sémantique faible pour le corpus : les prépositions, les nombres et enfin la ponctuation qui sort du cadre de notre analyse. Enfin, à l'aide du package de text-mining présent dans le logiciel R nous avons simplifié grammaticalement les mots afin de ne pas distingué les mots selon qu'ils soient pluriels ou singuliers, conjugués ou non pour les verbes... Ce type de réduction apparaît naturelle dans une analyse de type *bag of words*. L'étape suivante consiste en la création d'un dictionnaire qui associe de manière unique à chaque mot présent dans le corpus un unique nombre. Ce dictionnaire étant crée, nous avons pu transformer notre corpus textuel en une matrice numérique où chaque ligne correspond à un document et chaque case à un nombre qui représente un mot. Notre matrice possède donc 998 lignes, la taille de chaque ligne correspond à la taille du document et chaque éléments de la matrice est compris entre 1 et 2527 qui correspond à la taille du dictionnaire.

Analyse des résultats

Pour cette première implémentation, le choix du nombre de *topics* s'est fait par tâtonnement. Nous avons d'abord fixé le nombre de *topics* à $K = 2$, mais les résultats étaient peu cohérents au vu de la taille du corpus. Pour $K = 10$, l'information sur le regroupement des mots en *topics* était diffuse. En effet, pour un nombre de *topics* important les mots sont éparpillés, et il est alors difficile d'en cerner le sujet. Le choix optimal a été fixé à $K = 5$. L'algorithme d'inférence implémenté ici repose sur une méthode d'initialisation particulière. Comme nous l'avons vu, notre algorithme consiste en la résolution d'un problème d'optimisation non convexe. Pour cette raison, l'initialisation est un enjeu majeur pour obtenir des résultats satisfaisants. Nous avons donc décider d'initialiser nos paramètres α et β grâce à la méthode suivante. Nous tirons d'abord aléatoirement un nombre de documents du corpus (environ un dixième du corpus) et y implémentons l'algorithme sur ce nombre réduit de document. Les sorties α et β constituent alors les valeurs d'initialisation de l'inférence sur le corpus entier.

• **Analyse globale des *topics***

Le vecteur α représentant la répartition des cinq *topics* sur l'ensemble des mots des 998 documents du corpus vaut

$$\alpha = \begin{pmatrix} 0.234 & 0.133 & 0.168 & 0.335 & 0.442 \end{pmatrix}.$$

Les *topics* 3 et 4 semblent plus présents que les *topics* 2 et 3. De plus, le vecteur α permettant d'avoir une idée globale de la signification du corpus, on constate que le corpus de document mis à notre disposition est davantage représenté par les *topics* 3 et 4 (dont les coordonnées en α sont maximales).

L'algorithme calcule également la matrice β correspondant au poids de chaque mot du corpus dans les cinq *topics*. Cela permet donc d'associer à chaque mot un (voire plusieurs) *topic* et donc d'avoir une première idée des thématiques de chaque *topic*. En effet, en sélectionnant une ligne de la matrice β, on récupère la distribution des mots au sein du *topic* correspondant. Nous avons donc implémenté une fonction "Extraction" qui renvoie pour chaque *topic* les mots ayant la plus grande fréquence d'apparition.

Pour les 998 documents, les résultats sont consignés dans la Figure 13.

Topic 1 : Street	Topic 2 : Road Signs	Topic 3 : Car	Topic 4 : Vegetal Wastes
Street	Stop	Pothole	Trash
Light	Intersect	Widget	Pick
Streetlight	Side	Traffic	Tree
Map	Property	Park	Bulk
St	Sign	Car	Alley
Corner	Sidewalk	Front	Brush

FIGURE 13 – Les six mots les plus représentatifs pour chaque *topic* (par ordre d'importance)

Le *topic* 5 a été éliminé puisqu'il regroupait des mots ayant des significations différentes et qu'il était donc difficile à interpréter. On notera qu'il s'agit en réalité du *topic* le plus présent dans le corpus. En effet, les *topics* ayant le plus de sens sont moins représentés car ils regroupent des mots plus spécifiques, contrairement aux *topics* qui contiennent des mots aléatoires et dont le sujet est difficilement explicable.

• **Analyse spécifique des documents**

Il s'agit à présent d'étudier plus en détail les documents afin de dégager pour chacun d'eux le sujet dont il est question. La sortie ϕ de l'algorithme d'inférence est un *array* à 3 dimensions où est stocké l'ensemble des ϕ pour chacun des 998 documents. Pour un document d, la coordonnée de ϕ correspond à la probabilité d'avoir le *topic* z_n sachant le mot w_n pour α et β fixés. Pour cette analyse, nous nous restreindrons à deux documents, où les mots relatifs à un certain corpus seront colorés. Ensuite, nous les comparerons avec le vecteur γ indiquant la répartition des *topics* sur un même document afin de vérifier la cohérence des résultats.

Document 230 : "Street light glows bright yellow and then goes out. location - median of Grove half between Tuckahoe and Matoaka. North side of street."
On remarque une forte présence de mots appartenant au *topic* 1. Les résultats sont confirmés par le vecteur γ suivant :

$$\gamma = \begin{pmatrix} 6.859 & 2.671 & 0.164 & 1.263 & 1.263 \end{pmatrix}.$$

Document 285 : "Branches and tree debris located in the back next to the trash cans."
De la même manière, le document 285 semble aborder le *topic* 4 de par la présence de 6 mots sur 12 appartenant à ce dernier. Le vecteur γ présente une coordonnée pour le *topic* 4 égale à 7.254 tandis que les autres coordonnées

sont comprises entre 0.1 et 0.2.

L'intérêt du γ semble clair ; il permet d'assigner chaque document à un *topic*, alors que le vecteur α donne une répartition globale des *topics*. Le travail effectué sur le corpus nous a également permis de comprendre que les données étaient issues d'un forum de quartier où les habitants postent toutes les nouvelles du jour et incidents, notamment à propos des déchets et du trafic routier.

• Intérêt de l'étape d'initialisation

La méthode d'initialisation repose sur une compilation de l'algorithme sur des données réduites.
L'intérêt de cette méthode d'initialisation est d'ordre computationnel. En effet, nous avons remarqué empiriquement que l'implémentation de cette méthode réduit considérablement la complexité de l'algorithme avec des résultats plus satisfaisants.

$$\alpha = \begin{pmatrix} 1 & 1 & .. \end{pmatrix},$$

Sans cette méthode d'initialisation, en prenant par exemple une valeur de α mentionnée ci dessus et β tiré selon une loi de Dirichlet, nous avons rapidement constaté que la complexité de cette méthode était plus importante et que les résultats était moins satisfaisants. En effet, la méthode d'initialisation préalable sur un petit nombre de données permet de prendre en compte uniquement les mots apparaissant plusieurs fois, et fait abstraction des mots peu présents et dénués de sens. Le temps d'implémentation de l'algorithme en est alors considérablement réduit.

De plus, l'implémentation sans passer par l'étape d'initialisation nous a fourni des résultats peu convaincants. Les *topics* regroupaient des mots de significations diverses et variées, dont il était difficile de cerner le sujet. Le recours à cette étape d'initialisation nous a alors paru primordial au vu des résultats probants qu'elle offrait.

5.3 Applications

Nous avons choisis d'élargir notre étude dans deux directions.

La première est une application directe. Elle consiste en l'utilisation des différents paramètres α, β, γ et ϕ afin de trouver une technique de classification automatique de documents par thème. L'autre est plus complexe. Il s'agit d'un approfondissement de notre modèle LDA afin de mettre en place un algorithme de prédiction du nombre de vues par *post* sur nos données.

5.3.1 Classification

La première application est une interprétation presque immédiate de nos résultats. L'idée est de calculer une grandeur à partir de nos paramètres estimés α, β, γ et ϕ qui permettra de classifier les documents par *topics*. Ou plus précisemment, d'associer à chacun des documents un nombre mesurant l'importance du document pour un *topic* donné. Nous avons choisis intuitivement la grandeur suivante notée κ que l'on estime grâce au paramètre ϕ :

$$\kappa_{k,d} = \frac{E\left[\sum_n 1_{z_{n,d}=k}\right]}{N_d} = \frac{\sum_n \phi^*_{k,n,d}}{N_d}$$

Cela revient à considérer que pour un *topic* k et un document d donnés, plus il y a de mots associés au *topic* k, plus le document est fortement associé au *topic* considéré. Ainsi, en parcourant les documents et en sélectionnant la plus grande valeur de κ pour un *topic* k donné, on parvient à filtrer les documents selon les *topics*.

5.3.2 Prédiction de l'audience

Les données testées en Partie 5.2 sont issues d'un forum de discussion sur Internet. Une étude intéressante peut être d'essayer de prévoir les chances de réponse à un *post* sur le forum (un *post* correspondant dans notre étude à un document). L'idée va donc être d'enrichir notre modèle LDA afin d'y insérer une variable à expliquer Y qui mesure le nombre de vues du *post*. Excepté cette modification, le LDA présenté en Partie 4 reste identique.
On propose

$$Y_{d,k}|z \sim \mathcal{N}(\rho_k \sum_n 1_{z_n=k}, \sigma_k).$$

Cela signifie que le nombre de vues d'un *post* d sachant le *topic* k suit dans notre modèle une loi normale de variance inconnue propre au *topic* k et d'espérance la somme des mots du document associés au *topic* k, pondérée par un nombre propre au *topic* (la *"force"* du *topic*). Schématiquement, notre modèle bayésien devient le modèle expliqué en Figure 14.

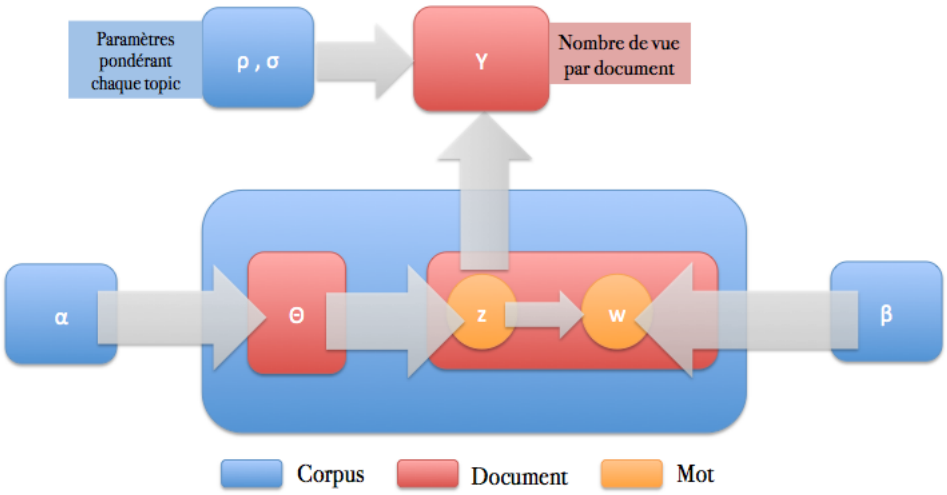


FIGURE 14 – LDA enrichi : Prévision du nombre de vues

L'objectif de l'inférence qui peut se faire de façon similaire serait à présent d'estimer ρ et σ. Une fois estimés, nous pourrions dire quelle loi suit "l'audience" d'un de nos *posts*. Ce prolongement du modèle LDA est intéressant car il nous fait passer d'un apprentissage non-supervisé (la découverte de *topic* d'un texte) à un apprentissage supervisé (présence d'une variable Y observée expliquée).

Nous nous sommes arrêtés à une étude purement théorique de ce modèle (et ne l'avons donc pas testé sur des données réelles).

Conclusion

Notre étude a donc consisté en la compréhension minutieuse, l'implèmentation et l'utilisation du modèle LDA. Pour résumer, le LDA consiste en des hypothèses portant sur la manière dont le langage est produit. Ces hypothèses simplificatrices au premier abord, capturent en réalité suffisamment d'information dans un but de découverte des thèmes abordés au sein de données textuelles.

Succinctement, ce modèle correspond à des distributions de probabilités qui relient les différentes composantes d'un corpus : des mots et des thèmes associés à ces mots, des documents et des répartitions de thèmes au sein de ces documents et enfin un corpus décris par des similarités de distribution des différents thèmes.

Ce modèle établi, l'idée suivante fut d'approcher la loi à posteriori capturant l'ensemble de l'information souhaitée. Dans ce but, nous avons mis en place une méthode déterministe d'optimisation qui couple un algorithme d'inférence variationnelle et d'estimation bayésienne afin d'approximer au mieux les différents paramètres du modèle sachant les données - les mots et la structure du corpus - et sachant les hypothèses propres au LDA.

Cette inférence faite, nous avons pu tester notre algorithme sur des données réelles afin d'en mesurer la puissance. Les résultats obtenus furent très satisfaisants. En particulier, ils nous ont permis de découvrir les différents thèmes abordés dans un corpus sans même le lire. Toutefois, ces résultats auraient pu être encore plus concluant, d'une part en nettoyant plus précisément les données de départ, et d'autre part en revoyant l'architecture de notre algorithme afin de gagner en efficacité. Ces deux modifications faites, nous aurions pu tester nos travaux sur des données plus volumineuses.

Non seulement très efficace, l'étude de ce modèle fut une excellente introduction aux modèles hierarchiques bayésiens. Ainsi, ces modèles apparaissent d'une richesse immense du point de vue de leur domaine d'application, l'approche bayésienne étant très flexible au sujet de la modélisation des données étudiées. En ce sens l'étude du langage n'est qu'un domaine d'application parmi d'autre, la reconnaissance vocale ou visuelle étant deux autres domaines d'études majeurs. En particulier, nous avons pu adapter le LDA dans une optique d'apprentissage supervisé afin de prévoir le nombre de vues d'un post.

Plus généralement, notre projet nous a permis d'appréhender la richesse de ce qu'on appelle désormais communément le *machine learning*. Cette discipline est à l'intersection de problématiques aussi diverses que la modélisation de l'information, le calcul de lois complexes par des méthodes d'optimisation ou de simulations, l'algorithmie et enfin la programmation. Dans cette perspective, l'étude du LDA en fut une parfaite illustration.

Références

Ricardo A. Baeza-Yates, Berthier Ribeiro-Neto.
Modern Information Retrieval, Addison-Wesley Longman Publishing Co., 1999.

Alberto Bietti.
Latent Dirichlet Allocation, 2012.
Christopher M. Bishop.
Pattern recognition and machine learning (Volume 1, Chapitre 10), Springer-Verlag New-York, 2006.

David M. Blei, Thomas L. Griffiths, Mark Steyvers, Joshua B. Tenenbaum.
Integrating Topics and Syntax, Advances in Neural Information Processing Systems, 2004.

David M. Blei, Andrex Y. Ng, Michael I. Jordan.
Latent Dirichlet Allocation (Volume 3, Pages 993 à 1022), The Journal of Machine Learning Research, 2003.

Stephen Boyd, Lieven Vandenberghe.
Convex Optimization, Cambridge University Press, 2004.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Richard Harshman, Thomas K. Landauer.
Indexing by Latent Semantic Analysis, Journal of the American Society of Information Science, 1990.

Tom Griffiths.
Gibbs sampling in the generative model of Latent Dirichlet Allocation, 2002.

Thomas Hofmann.
Probabilistic latent semantic indexing (Pages 50 à 57), SIGIR '99, Proceedings of the 22^{nd} annual international ACM SIGIR conference on research and development in information retrieval, 1999.

Gerard Salton, Michael J. McGill.
Introduction to Modern Information Retrieval, McGraw-Hill, Inc., 1986.

Annexes

Annexe 1 : Mélange de gaussiennes : calcul des distributions marginales

On reprend les résultats de la Partie 3 sur la loi jointe de (X, Z, θ) :

$$\begin{aligned} p(X,Z,\theta) &\propto p(\theta)\cdot\prod_{i=1}^{N} p(X_i|Z_i,\theta)\cdot p(Z_i), \\ p(X,Z,\theta) &\propto \phi(\theta,0,1)\cdot\prod_{i=1}^{N}\left[\phi(X_i,0,1)^{1_{Z_i=1}}\cdot\phi(X_i,\theta,1)^{1_{Z_i=0}}\right]\left[p^{1_{Z_i=1}}\cdot(1-p)^{1_{Z_i=0}}\right]. \end{aligned}$$

On essaie donc d'obtenir les distributions marginales des variables latentes Z_i et du paramètre θ.

Calcul de la distribution marginale de Z_i

Alors la distribution marginale de Z_i s'écrit, à p fixé :

$$\begin{aligned} q_i(z_i) &\propto e^{\,E_{\theta,Z_{j(j\neq i)}}\ln\left[[p\cdot\phi(X_i,0,1)]^{1_{Z_i=1}}\cdot[(1-p)\cdot\phi(X_i,\theta,1)]^{1_{Z_i=0}}\right]}, \\ q_i(z_i) &\propto [p\cdot\phi(X_i,0,1)]^{1_{Z_i=1}}\cdot(1-p)^{1-Z_i}\cdot e^{\ln E_{\theta,Z_{j(j\neq i)}}[\phi(X_i,\theta,1)]^{1_{Z_i=0}}}, \\ q_i(z_i) &\propto [p\cdot\phi(X_i,0,1)]^{Z_i}\cdot\left[\frac{1-p}{\sqrt{2\pi}}\cdot e^{-\frac{E_\theta(X_i-\theta)^2}{2}}\right]^{1-Z_i}. \end{aligned}$$

On reconnaît ici l'expression d'une loi de Bernoulli. Après normalisation, on obtient donc p_i la probabilité que la variable latente Z_i prenne la valeur 1 :

$$p_i = \frac{p\cdot\phi(X_i,0,1)}{p\cdot\phi(X_i,0,1)+\frac{1-p}{\sqrt{2\pi}}\cdot e^{-\frac{E_\theta(X_i-\theta)^2}{2}}}. \tag{4}$$

Calcul de la distribution marginale de θ

On s'intéresse à présent à la distribution marginale de θ. Par la même formule que pour la distribution marginale de Z_i, on obtient :

$$\begin{aligned} q_\theta(\theta) &\propto \phi(\theta,0,1)\cdot\prod_{i=1}^{N}[(1-p)\cdot\phi(X_i,\theta,1)]^{E_{Z_i}[1_{Z_i=1}]}, \\ q_\theta(\theta) &\propto \frac{1}{\sqrt{2\pi}}\cdot e^{-\frac{\theta^2}{2}}\cdot\prod_{i=1}^{N}\frac{1-p}{\sqrt{2\pi}}\cdot e^{-\frac{(1-p_i)\cdot(X_i-\theta)^2}{2}}, \\ q_\theta(\theta) &\propto e^{-\frac{\theta^2}{2}-\sum_i\frac{(1-p_i)\cdot(X_i-\theta)^2}{2}}, \\ q_\theta(\theta) &\propto e^{-\frac{1}{2}\cdot\left[\theta^2+\sum_i((1-p_i)X_i^2-2(1-p_i)X_i\theta+(1-p_i)\theta^2)\right]}, \\ q_\theta(\theta) &\propto e^{-\frac{1}{2}\cdot\left[(1+\sum_i(1-p_i))\theta^2-2(\sum_i(1-p_i)X_i)\cdot\theta\right]}, \\ q_\theta(\theta) &\propto e^{-\frac{1}{2}\cdot\frac{\left(\theta-\frac{\sum_i(1-p_i)X_i}{1+\sum_i(1-p_i)}\right)^2}{\frac{1}{1+\sum_i(1-p_i)}}}. \end{aligned}$$

On reconnaît ici l'expression d'une loi normale. En notant

$$\begin{cases} m = \frac{\sum_i (1-p_i) X_i}{1+\sum_i (1-p_i)}, \\ \quad \text{et} \\ \sigma^2 = \frac{1}{1+\sum_i (1-p_i)}, \end{cases} \tag{5}$$

q_θ est une loi normale de moyenne m et de variance σ^2.

Retour sur le calcul de la distribution marginale de Z_i

Grâce à cette expression de q_θ, on peut calculer $E_\theta[(X_i - \theta)^2]$ pour obtenir p_i.

$$E_{q_\theta}\left[(X_i-\theta)^2\right] = E_{q_\theta}\left[(X_i-m)^2\right] + E_{q_\theta}\left[(m-\theta)^2\right] + 2E_{q_\theta}\left[(X_i-m)(m-\theta)\right],$$

$$E_{q_\theta}\left[(X_i-m)^2\right] = (X_i-m)^2,$$

$$E_{q_\theta}\left[(m-\theta)^2\right] = \sigma^2,$$

$$\begin{aligned} E_{q_\theta}\left[(X_i-m)(m-\theta)\right] &= E_{q_\theta}\left[m \cdot X_i\right] - E_{q_\theta}\left[m^2\right] - E_{q_\theta}\left[\theta \cdot X_i\right] + E_{q_\theta}\left[m \cdot \theta\right] \\ &= m \cdot X_i - m^2 - X_i \cdot E_{q_\theta}\left[\theta\right] + m \cdot E_{q_\theta}\left[\theta\right] \\ &= m \cdot X_i - m^2 - m \cdot X_i + m^2 \\ &= 0, \end{aligned}$$

$$E_{q_\theta}\left[(X_i-\theta)^2\right] = (X_i-m)^2 + \sigma^2.$$

Résultat

Finalement, en reprenant les expressions de m et de σ^2, on obtient les lois marginales suivantes pour le paramètre θ et les variables latentes Z_i :

$$\begin{cases} q_\theta \sim \mathcal{N}(m, \sigma^2), \\ q_{Z_i} \sim \mathcal{B}\left(\frac{p \cdot \phi(X_i,0,1)}{p \cdot \phi(X_i,0,1) + \frac{1-p}{\sqrt{(2\pi)}} * e^{-\frac{(X_i-m)^2+\sigma^2}{2}}} \right). \end{cases} \tag{6}$$

Annexe 2 : Implémentation du mélange de gaussiennes sous R

Voici le code que nous avons utilisé pour tester les méthodes d'inférence variationnelle sur un mélange de gaussiennes. On rappelle qu'on a appelé p_i la probabilité que la variable latente Z_i prenne la valeur 1, et m et σ^2 respectivement la moyenne et la variance du paramètre θ, qui suit une loi normale.

```
## On commence par créer une fonction gen_donnees qui génère des données selon le mélange
gaussien décrit en Partie 3, à partir du nombre de données souhaitées n, de la probabilité
de suivre une gaussienne centrée réduite prob et du paramètre  θ qui est
l'espérance de la seconde gaussienne du mélange.
gen_donnees <- function(n,prob,theta){
  U=rbinom(n,1,prob)
  W=rnorm(n,0,1)
  V=rnorm(n,theta,1)
  X=U*W+(1-U)*V
  return(X)
}


## Puis on crée la fonction varbayes, qui prend en argument les données X, la probabilité
que X suive une gaussienne centrée réduite prob, et le pas souhaité. Le pas est le critère
d'arrêt de l'algorithme ; lorsque la différence absolue entre les moyennes m obtenues après
deux itérations successives de l'algorithme sera inférieure au pas, l'algorithme terminera.
La terminaison de l'algorithme est garantie par la convergence des m, d'après la théorie de
la Partie 2.
varbayes <- function(X,prob,pas)

  ## Initialisation des valeurs
  n=length(X)
  ## Connaître la taille de X permet de savoir combien de pi il nous faudra calculer.
  p0=rep(0.5,n)
  ## Le vecteur p0 contient les probabilités initiales des Xi de suivre une gaussienne
  centrée réduite, arbitrairement fixées à 0,5. C'est initialement un vecteur de taille n.
  Mais à chaque itération, il sera concaténé avec le vecteur p défini ci-dessous. Il gardera
  donc en mémoire  toutes les valeurs des pi qui auront été calculées pour chaque Xi à chaque
  itération. Cela n'est pas indispensable, puisque seules les pi finales importent, mais cela
  nous permet d'étudier la convergence de l'algorithme sur de petites quantités de données.
  p=rep(0,n)
  ## Le vecteur p contiendra à chaque itération les nouvelles probabilités pi qui auront
  été calculées. C'est un vecteur de taille n.
  m=c()
  m[1]=10
  ## Le vecteur m contiendra toutes les valeurs de m calculées à chaque itération de
  l'algorithme. C'est un vecteur de taille "le nombre d'itérations" + 1.
  sigma=c()
  sigma[1]=1
  ## Le vecteur sigma contiendra toutes les valeurs de σ2
  calculées à chaque itération de  l'algorithme. C'est un vecteur de taille
  "le nombre d'itérations" + 1.

  ## Nous allons maintenant mettre en place une boucle while pour itérer les calculs de m,          σ2
  variable evolution indique la valeur absolue de la différence entre la valeur de m à
  l'itération précédente et à l'itération en cours, qui est notre critère d'arrêt. Pour
  être certains d'entrer dans la boucle while, cette valeur sera initialisée ) (pas+1).
  j=2
  evolution=pas+1

  while (evolution>pas)
    m[j]=sum(X*(1-p))/(1+sum(1-p))
```

```
    sigma[j]=1/(1+sum(1-p))
    for (i in 1:n)
      p[i]=(prob*dnorm(X[i],0,1))/
      ((1-prob)*(1/sqrt(2*pi))*exp(-0.5*((X[i]-m[j])^2+sigma[j]))+prob*dnorm(X[i],0,1))

    p0=cbind(p0,p)
    ## On concatène p0 et p
    evolution=abs(m[j]-m[j-1])
    j=j+1
    ## On actualise a valeur de la variable evolution et de j.

  resultat <- data.frame(m,sigma,t(p))
  ## Le résultat sera l'ensemble des valeurs prises par m, sigma et les pi au cours
  de l'algorithme.
  return(t(resultat))


p=0.2
## On choisit p la probabilité que X suive une gaussienne centrée réduite.

X=gen_donnees(1000,p,3)
## On génère 1000 données tirées selon le mélange gaussien décrit dans la Partie 3,
avec theta=3.

VB <- varbayes(X,p,0.001)
## On applique l'algorithme de variational Bayes aux données générées, avec un
critère d'arrêt de 0.001.


## On affiche d'abord les données qui ont été générées.
plot(X, main = "", xlab = "Numéro d'observation", ylab = "x", type = "p", col = "green")

## On affiche ensuite la densité estimée des données générées.
dX <- density(X)
plot(dX,main = "", xlab = "", ylab = "Densité")
polygon(dX, col="green", border="chartreuse4")

## Convergence de m
plot(2:7,VB[1,2:7],main="",xlab="Numéro de l'itération",ylab="m",type="b",col="green")
VB[1,7] ##2.970889

## Convergence de sigma
plot(2:7,VB[2,2:7],main="",xlab="Numéro de l'itération",ylab="sigma2",type="b",col="green")
VB[2,7] ##0.001264566

## Convergence des pi
X[1] ##4.262318
plot(2:7,VB[3,2:7],main="",xlab="Numéro de l'itération",ylab="p1",type="b",col="green")
VB[3,7] ##6.536583e-05

X[2] ## -1.963522
plot(2:7,VB[4,2:7],main="",xlab="Numéro de l'itération",ylab="p2",type="b",col="green")
VB[4,7] ## 0.9998693
```

```
## pi finaux en fonction des Xi
plot(X,VB[3:dim(VB)[1],dim(VB)[2]],main="",xlab="X",ylab="pi",col="green")

## Densité des pi finaux
dp<-density(VB[3:dim(VB)[1],dim(VB)[2]])
plot(dp,main="",xlab="",ylab="Densité",col="green")
polygon(dp, col="green", border="chartreuse4")

## Xi colorés en fonction de la valeur de pi
color<-c(length(X))
color[VB[3:dim(VB)[1],dim(VB)[2]]>0.5]<-"red"
color[VB[3:dim(VB)[1],dim(VB)[2]]<=0.5]<-"blue"
plot(X,main="",xlab="Numéro de l'observation",ylab="X",col=color)
```

Annexe 3 : Calcul de la vraisemblance du modèle

$$\begin{aligned} L(\gamma,\phi|\alpha,\beta) &= E_{q(\theta,z)}\left[log\, p(\theta,z,w|\alpha,\beta)\right] - E_{q(\theta,z}\left[log\, q(\theta,z)\right] \\ &= \underbrace{E_{q(\theta,z)}[log\, p(\theta|\alpha)]}_{1} + \underbrace{E_{q(\theta,z)}[log\, p(z|\theta,\alpha)]}_{2} + \underbrace{E_{q(\theta,z)}[log\, p(w|z,\beta)]}_{3} - \underbrace{E_{q(\theta)}[log\, q(\theta)]}_{4} - \underbrace{E_{q(z)}[log\, q(z)]}_{5} \end{aligned}$$

Calcul du premier terme

$$p(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{K}\alpha_i)}{\prod_{i=1}^{K}\Gamma(\alpha_i)} \cdot \theta_1^{\alpha_1-1} \cdot \ldots \cdot \theta_K^{\alpha_K-1}.$$

On en déduit que :

$$\begin{aligned} E_{q(\theta,z)}[log\, p(\theta|\alpha)] &= log\,\Gamma\left(\sum_{i=1}^{K}\alpha_i\right) - \sum_{i=1}^{K} log\,\Gamma(\alpha_i) + E_{q(\theta,z)}\left[\sum_{i=1}^{K}(\alpha_i-1)\cdot log\,\theta_i|\gamma\right] \\ &= log\,\Gamma\left(\sum_{i=1}^{K}\alpha_i\right) - \sum_{i=1}^{K} log\,\Gamma(\alpha_i) + \sum_{i=1}^{K}(\alpha_i-1)\cdot E_{q(\theta)}\left[log\,\theta_i|\gamma\right] \\ &= log\,\Gamma\left(\sum_{i=1}^{K}\alpha_i\right) - \sum_{i=1}^{K} log\,\Gamma(\alpha_i) + \sum_{i=1}^{K}(\alpha_i-1)\cdot\left[\psi(\gamma_i) - \psi\left(\sum_{j=1}^{K}\gamma_j\right)\right] \end{aligned}$$

La dernière égalité vient des propriétés des modèles exponentiels. Dans le modèle de Dirichlet, le paramètre naturel est $\eta_i = \alpha_i - 1$ et la statistique suffisante est $T(\theta_i) = log\,\theta_i$. ψ étant la log-dérivée de la loi Γ, on en déduit que :

$$E_{q(\theta)}\left[log\,\theta_i|\gamma\right] = \psi(\gamma_i) - \psi\left(\sum_{j=1}^{K}\gamma_j\right)$$

Calcul du deuxième terme

$$
\begin{aligned}
E_{q(\theta,z)}[log\, p(z|\theta,\alpha)] &= \sum_{n=1}^{N} E_{q(\theta,z)}[log\, p(z_n|\theta,\alpha)] \\
&= \sum_{n=1}^{N} E_{q(\theta,z)}\left[\sum_{i=1}^{K} z_{ni} \cdot log\, \theta_i\right] \\
&= \sum_{n=1}^{N} E_{q(\theta)}\left[\sum_{z_n}\sum_{i=1}^{K} z_{ni} \cdot log\, \theta_i \cdot q(z_n|\phi_n)\right] \\
&= \sum_{n=1}^{N} E_{q(\theta)}\left[\sum_{i=1}^{K}\sum_{z_n} z_{ni} \cdot log\, \theta_i \cdot \underbrace{q(z_n|\phi_n)}_{=\phi_{nm}/z_{ni}=1 \text{ ssi } i=m}\right] \\
&= \sum_{n=1}^{N} E_{q(\theta)}\left[\sum_{i=1}^{K} \phi_{ni} \cdot log\, \theta_i\right] \\
&= \sum_{n=1}^{N}\sum_{i=1}^{K} E_{q(\theta)}\left[\phi_{ni} \cdot log\, \theta_i\right] \\
&= \sum_{n=1}^{N}\sum_{i=1}^{K} \phi_{ni} \cdot E_{q(\theta)}\left[log\, \theta_i|\gamma\right] \\
&= \sum_{n=1}^{N}\sum_{i=1}^{K} \phi_{ni} \cdot \left[\psi(\gamma_i) - \psi\left(\sum_{j=1}^{K}\gamma_j\right)\right]
\end{aligned}
$$

Calcul du troisième terme

$$
\begin{aligned}
E_{q(\theta,z)}[log\, p(w|z,\beta)] &= \sum_{n=1}^{N} E_{q(z_n)}\left[log\, p(w_n|z_n,\beta_n)\right] \\
&= \sum_{n=1}^{N} E_{q(z_n)}\left[\sum_{j=1}^{V} log(\beta_{z_n j}^{w_{nj}})\right] \\
&= \sum_{n=1}^{N}\sum_{z_n}\sum_{j=1}^{V} w_{nj} \cdot log\, \beta_{z_n j} \cdot \underbrace{q(z_n|\phi_n)}_{=\phi_{nm}/z_{ni}=1 \text{ ssi } i=m} \\
&= \sum_{n=1}^{N}\sum_{z_n}\sum_{j=1}^{V} w_{nj} \cdot log\, \beta_{ij} \cdot \phi_{ni}
\end{aligned}
$$

Calcul du quatrième terme

$$\begin{aligned}
E_{q(\theta)}[log\, q(\theta)] &= E_{q(\theta)}\left[log\left(\frac{\Gamma(\sum_{i=1}^{K}\gamma_i)}{\prod_{i=1}^{K}\Gamma(\gamma_i)}\cdot\theta_1^{\gamma_1-1}\cdot\ldots\cdot\theta_K^{\gamma_K-1}\right)\right] \\
&= log\,\Gamma\left(\sum_{i=1}^{K}\gamma_i\right) - \sum_{i=1}^{K} log\,\Gamma(\gamma_i) + \sum_{i=1}^{K}(\gamma_i-1)\cdot E_{q(\theta)}[log\,\theta_i] \\
&= log\,\Gamma\left(\sum_{i=1}^{K}\gamma_i\right) - \sum_{i=1}^{K} log\,\Gamma(\gamma_i) + \sum_{i=1}^{K}(\gamma_i-1)\cdot\left[\psi(\gamma_i) - \psi\left(\sum_{j=1}^{K}\gamma_j\right)\right]
\end{aligned}$$

Calcul du cinquième terme

$$\begin{aligned}
E_{q(z)}[log\, q(z)] &= E_{q(z)}\left[\sum_{n=1}^{N} log\, q(z_n|\phi_n)\right] \\
&= \sum_{n=1}^{N} E_{q(z)}\left[\sum_{i=1}^{K} log\,\phi_{ni}\cdot z_i\right] \\
&= \sum_{n=1}^{N}\sum_{z_n}\sum_{i=1}^{K} z_{ni}\cdot log\,\phi_{ni}\cdot \underbrace{q(z_n|\phi_n)}_{=\phi_{nm}/z_{ni}=1 \text{ ssi } i=m} \\
&= \sum_{n=1}^{N}\sum_{i=1}^{K}\phi_{ni}\cdot log\,\phi_{ni}
\end{aligned}$$

Conclusion sur le calcul de la vraisemblance

On en déduit ainsi la formule de $L(\gamma,\phi;\alpha,\beta)$:

$$\begin{aligned}
L(\gamma,\phi;\alpha,\beta) &= log\,\Gamma\left(\sum_{i=1}^{K}\alpha_i\right) - \sum_{i=1}^{K} log\,\Gamma(\alpha_i) + \sum_{i=1}^{K}(\alpha_i-1)\cdot\left[\psi(\gamma_i) - \psi\left(\sum_{j=1}^{K}\gamma_j\right)\right] \\
&+ \sum_{n=1}^{N}\sum_{i=1}^{K}\phi_{ni}\cdot\left[\psi(\gamma_i) - \psi\left(\sum_{j=1}^{K}\gamma_j\right)\right] \\
&+ \sum_{n=1}^{N}\sum_{z_n}\sum_{j=1}^{V} w_{nj}\cdot log\,\beta_{ij}\cdot\phi_{ni} \\
&- log\,\Gamma\left(\sum_{i=1}^{K}\gamma_i\right) + \sum_{i=1}^{K} log\,\Gamma(\gamma_i) - \sum_{i=1}^{K}(\gamma_i-1)\cdot\left[\psi(\gamma_i) - \psi\left(\sum_{j=1}^{K}\gamma_j\right)\right] \\
&- \sum_{n=1}^{N}\sum_{i=1}^{K}\phi_{ni}\cdot log\,\phi_{ni}
\end{aligned}$$

Annexe 4 : Code R de génération d'un corpus de textes

```
setwd("C:/Users/Chemin")
library(gtools)

## as.numeric(dictionnaire_liste["mot"]) renvoie l'indice du mot "mot"
dictionnaire_liste <- list(amour=1, bonheur=2, chômage=3, confiance=4, employeur=5, salaire=6)
theme_bonheur<-c("amour","bonheur","confiance")
theme_travail<-c("chômage","employeur","salaire")
## dictionnaire_vect[a] renvoie le mot du dictionnaire numéroté a
dictionnaire_vect <- sort(c(theme_bonheur,theme_travail))
## On sauvegarde
write.csv(dictionnaire_vect, file = "vect_dictionnaire.csv")

## On choisit le nombre de documents n.
n=50

## On appelle V le nombre de mots du dictionnaire.
V=length(dictionnaire_liste)

## On a deux topics, donc alpha est de dimension 2. On choisit alpha[1] très grand devant alpha[2].
alpha <- c(3,1)

## On a V mots et 2 topics, donc beta est une matrice de dimension Vxk.
## On remplit la première ligne de beta avec les probabilités que, lorsqu'on est dans le
thème du bonheur, les mots de theme_bonheur apparaissent.
generation_beta=function(V,dictionnaire_liste)
  beta <- matrix(0,nrow=2,ncol=V)
  beta[1,as.numeric(dictionnaire_liste["amour"])]=2
  beta[1,as.numeric(dictionnaire_liste["bonheur"])]=3
  beta[1,as.numeric(dictionnaire_liste["confiance"])]=0.25
  beta[2,as.numeric(dictionnaire_liste["chômage"])]=2
  beta[2,as.numeric(dictionnaire_liste["salaire"])]=3
  beta[2,as.numeric(dictionnaire_liste["employeur"])]=0.25
  #On normalise.
  beta[1,]<-beta[1,]/sum(beta[1,])
  beta[2,]<-beta[2,]/sum(beta[2,])
  return(beta)

## On génère beta.
beta = generation_beta(V,dictionnaire_liste)
## On sauvegarde
write.csv(beta, file = "beta.csv")


## On choisit xhi pour pouvoir tirer le nombre de mots d'un texte.
xhi=40

## Créons un vecteur taille_documents qui donnera la taille des documents.
taille_documents <- rpois(n,xhi)

## Créons un vecteur theta_documents qui donnera les topics de chaque document.
theta_documents <- rdirichlet(n, alpha)
```

```
## Taille du plus grand document.
Nmax=max(taille_documents)

## On crée une matrice qui indiquera pour chaque mot du corpus à quel topic il appartient.
generation_topics_mots <- function(n,Nmax,taille_documents)
  Z <- matrix(0,nrow=n,ncol=Nmax)
  for (i in 1:n)
    for (j in 1:taille_documents[i])
      #On choisit Z selon la multinomiale(theta du document)
      Z[i,j] = sum(rmultinom(1, 1, theta_documents[i,])*c(1,2))


  return(Z)

## On génère Z
Z = generation_topics_mots(n,Nmax,taille_documents)
## On sauvegarde
write.csv(Z, file = "topics.csv")


## Créons une matrice qui sera notre corpus. Chaque ligne sera un document. Remplissons-le.
generation_corpus <- function(n,Nmax,theta_documents,Z)
  Corpus <- matrix(0,nrow=n,ncol=Nmax)
  for (i in 1:n)
    for (j in 1:taille_documents[i])
      Corpus[i,j] = sum(rmultinom(1,1,beta[Z[i,j],])*c(1:V))


  return(Corpus)

## On génère Corpus
Corpus=generation_corpus(n,Nmax,theta_documents,Z)
## On sauvegarde
write.csv(Corpus, file = "corpus.csv")


## On remplit un corpus qui contient des mots et non des chiffres
generation_corpus_de_mots <- function(Corpus,dictionnaire_vect)
  Corpus_mots <- matrix("",nrow=dim(Corpus)[1],ncol=dim(Corpus)[2])
  for (i in 1:n)
    for (j in 1:taille_documents[i])
      Corpus_mots[i,j] = dictionnaire_vect[Corpus[i,j]]


  return(Corpus_mots)

## On génère le corpus de mots
Corpus_mots = generation_corpus_de_mots(Corpus,dictionnaire_vect)
## On sauvegarde
write.csv(Corpus_mots, file = "corpus_mots.csv")
```

```
## On affiche le Corpus créé
affichage_corpus_mots = function(Corpus_mots)
  texte=rep("",n)
  for (i in 1:n)
    for (j in 1:taille_documents[i])
      texte[i] <- paste(texte[i],Corpus_mots[i,j], sep = " ", collapse = NULL)


  return(texte)

## On affiche le Corpus
aff_corpus_mots <- affichage_corpus_mots(Corpus_mots)
## On sauvegarde
write.csv(aff_corpus_mots, file = "affichage_corpus_mots.csv")
```

Annexe 5 : Autres simulations de données

Simulation d'un corpus avec β uniformément distribué et

$$\alpha = \begin{pmatrix} 1 & 3 \end{pmatrix}$$

Paramètres du corpus		**Alpha**					
	Bonheur	3					
	Travail	1					
	Beta	Amour	Bonheur	Chômage	Confiance	Employeur	Salaire
	Bonheur	0.33	0.33	0	0.33	0	0
	Travail	0	0	0.33	0	0.33	0.33
Approximation		**Alpha**					
	Bonheur	2.37					
	Travail	0.85					
	Beta	Amour	Bonheur	Chômage	Confiance	Employeur	Salaire
	Bonheur	0.31	0.34	0.06	0.32	0.01	0.001
	Travail	0.06	0.02	0.32	0.03	0.24	0.31

FIGURE 15 – Comparaison des paramètres et résultats