A la recherche d'une classification de problèmes de mathématiques scolaires : une chronique

Luc-Olivier Pochon, Alain Favre

A la recherche d'une classification de problèmes de mathématiques scolaires : une chronique

L-O. Pochon et A. Favre

Résumé : Cette chronique retrace quelques essais pour utiliser des analyses à variables latentes pour classer des problèmes mathématiques.

Mot-clés : banque des données, problèmes de mathématiques, LSA, pLSA, LDA.

1. Le problème

Comment classer des problèmes de mathématiques scolaires afin qu'ils soient facilement accessibles aux enseignants sur Internet ? Tel est le questionnement à la base de ce travail. Les sites proposant des problèmes mathématiques sont nombreux mais leur utilisation n'est pas toujours aisée. Il manque un moteur de recherche spécialisé dans ce domaine. Un tel dispositif est-il envisageable ?

Pour réfléchir à cette problématique de façon pratique nous nous sommes proposés d'utiliser la banque de problèmes constituée par l'association du Rallye mathématique transalpin[1]. Cette banque de problèmes[2] rassemble les problèmes proposés à de nombreuses classes depuis une vingtaine d'années. Elle recense actuellement plus de 2000 problèmes.

Le jeu (corpus) de données offert par cette banque va nous servir à deux apprentissages. Le premier, le nôtre, aux nouvelles techniques et algorithmes de classification. Le deuxième concerne l'apprentissage d'un modèle de « problème de mathématiques ».

De nombreuses pistes complémentaires méritent d'être explorées. Certaines relèvent de méthodes statistiques pour l'analyse de textes. D'autres se réfèrent à des analyses symbolico-conceptuelles qui passent par des techniques ontologiques et de réseaux sémantiques que nous avons déjà explorées par ailleurs (Pochon, 2006, 2007).

Ce document, sorte de chronique, relate l'exploration de la première piste. Il présentera rapidement la banque de problèmes utilisée, les questions à traiter et les objectifs de l'étude. La partie suivante est consacrée aux analyses avec en préalable le pré-traitement des données brutes, c'est-à-dire leur mise en forme pour leur utilisation dans les l'analyses ultérieures. La conclusion fera état des problèmes rencontrés et du programme pour le travail futur.

2. La banque de problèmes du RMT et les questions

A la base de la banque des problèmes, il y a le travail de création des problèmes par des équipes internationales, de leur passation, de leur correction et de leur analyse. Au départ, chaque problème fait l'objet d'une fiche constituant les données brutes fournies par les concepteurs des problèmes. Il s'agit du titre du problème, de son énoncé et sa consigne, de l'analyse *a priori*, celle-ci se composant de l'analyse (a priori) de la tâche et du domaine mathématique, et des critères de correction. Le domaine mathématique concerne les concepts mathématiques mis en oeuvre et parfois des informations sur les « tâches » à mener (par exemple : décomposition et recomposition de surfaces en triangles). Par la suite les problèmes les plus intéressants font l'objet d'une analyse *a posteriori* qui peut être publiée dans la Gazette de Transalpie ou d'autres revues.

Les fiches informatisées complétées comportent principalement une identification de localisation (code d'identification, numéros du rallye, de l'épreuve et du problèmes), des *éléments de*

1 http://www.armtint.org/

2 http://www.projet-ermitage.org/ARMT/

classification (domaines de mathématiques et familles de tâches) puis les *données* proprement dites : titre, énoncé, et consigne du problème, analyse de la tâche, résumé de la tâche, concepts mathématiques mis en oeuvre (ci-après mot-clés). Finalement des *compléments* concernent les résultats, les erreurs commises, etc. Ils ne sont pas utilisés dans ce travail.

La classification adoptée pour les fiches informatisées demandent de traiter quatre questions principales. La première concerne l'adéquation des éléments de classification aux données. La deuxième concerne le degré de redondance entre les différents critères proposés. C'est chaque fois un problème de cohérence qui se pose et différentes techniques d'analyse pourront être envisagées pour ce travail qui fera l'objet d'un autre compte-rendu.

Estimer la possibilité d'automatiser, voire de remplacer, la classification manuelle par tâches constitue la troisième question. Finalement, il s'agit de contrôler si les critères de recherche issu des classifications sont utilisables par les usagers.

La troisième question nécessite un éclaircissement. Dans le cadre d'un enseignement mathématique basé sur la résolution de problèmes, la classification par tâches revêt une grande importance. Dans cette perspective, il ne suffit pas de donner la liste des notions mathématiques en présence, mais également les actions ou tâches qui les mettent à l'oeuvre. Dans le cas de la banque de problèmes du RMT cette classification se révèle délicate et n'est pas entièrement stabilisée. La description de certaines tâches est ambigüe et comprise différemment par les utilisateurs. Certaines tâches sont aussi équivalentes, d'autres mériteraient d'être décomposées en sous-tâches, etc.[1] Le traitement automatique sur une partie stable de la banque pourrait servir à contrôler, rectifier ou encore uniformiser la classification de l'ensemble de la banque.

Les deux dernières questions sont deux applications classiques (classification, restitution) de l'analyse de texte (Gimpel, 2006) auxquels pourra s'ajouter le problème de la mise à jour de la classification par ajout de données. Elles font l'objet de cette chronique.

Ce travail part donc de l'hypothèse que les tâches sont des variables cachées enfuies dans les énoncés, les analyses, résumés, etc. Les techniques d'extraction de ces variables cachées, thèmes, sujets ou *topics* dans la littérature en anglophone, semblent donc adaptées pour répondes à nos deux questions principales qui s'opérationnalisent en trois objectifs :

- Comparer différentes méthodes de réduction-classification et d'apprécier leur difficulté de mise en oeuvre

- Evaluer de façon sommaire quelle méthode semble la plus fidèle. Préalablement, les critères d'évaluation sont à examiner. Cet examen est à prolonger à des documents hors du corpus et donc concerne l'opération de récupération (information retrieval, restitution) puisque une requête peut-être considérée comme un nouveau document.

- Constater la possibilité d'une mise en relation de la classification par tâches et par *topics*.

L'étude présente s'occupe principalement des deux premiers objectifs. Le troisième fera l'objet d'un autre article.

3. Les données et leur pré-traitement

Les informations concernant la tâche sont contenues dans l'analyse de la tâche, les consignes et le résumé. Par ailleurs, les mot-clés renseignent sur les notions mathématiques. Ces données brutes nécessitent d'être triées et uniformisées afin de constituer des sacs de mots[2].

1 Pour les problèmes rencontrés, voir Pochon (2019).

2 Les données du problème paraissent moins utiles dans la mesure où elles présentent des situations où les mathématiques ne transparaissent pas ou peu.Nous nous réservons la possibilité de les traiter ultérieurement.

Il existe de nombreux systèmes permettant d'uniformiser les données textuelles. Des packages Python ou R proposent des outils de pré-traitement des ressources textuelles à analyser (voir annexe 1 : Parsing et chunking). Ces outils, la plupart adaptés à la langue anglaise, procèdent généralement à une « lemmatisation »[1] suivie d'une étape de « stemmatisation », ou « racinisation », ou encore désuffixation[2]. Il peut s'agir d'algorithmes sophistiqués qui recherchent la racine des mots pour unifier, par exemple, les idées que peuvent représenter un verbe et le nom associé. Parfois, c'est une simple troncature qui est adoptée.

Pour mieux contrôler cette étape en étant plus proche du domaine concerné, nous avons utilisé nos propres procédures en reprenant des travaux effectués dans le cadre des projets Prof'Expert et Ermitage[3]. En effet, entrer dans les usines à gaz proposées par certains packages n'est pas une mince affaire pour un résultat pas toujours convaincant.

Ces procédures couvrent quatre étapes :

3.1 Extraction des données de la banque

Il s'agit d'une procédure (en *php*) qui extrait dans un fichier texte les différents champs des fiches de la banque de problèmes : éléments de classification, mot-clés, résumé, énoncé, consigne, analyse de tâche.

Les trois étapes suivantes sont programmées en Prolog[4].

3.2. Tokenisation

Cette opération classique est accompagnée d'une normalisation des caractères (en transformant par exemple les codes *nfd* en *nfc*). En effet, les données de la banque sont souvent produites par couper-coller à partir de documents dont les encodages peuvent être divers. Les tokens considérés sont les mots, les nombres, les signes de ponctuation et les différents symboles utilisés en mathématiques.

3.3. Analyse grammaticale (parsing)

Le *parser* procède à une analyse grammaticale et crée une database prolog de quadruplets (code du document, champ concerné, foncteur et mot concerné, mot lemmatisé). Les mots au pluriel sont donc remplacés par le singulier et les verbes conjugués par l'infinitif. Par exemples :

```
item("op95",resum,nom_adj("naturels"),"naturel"),

item("op95",tache,verbe("disant"),"dire")
```

Les mots outil (déterminants, conjonctions, etc.) sont éliminés dans l'opération. Les formules et autres formalismes mathématiques sont aussi momentanément éliminés.

Certains champs sont particuliers : les éléments de classification qui ne sont pas à normaliser (par exemple : `item("op100",ident,null,"OPZ"))` et les mot-clés qui peuvent être constitués de plusieurs mots (exemple : `item("fn5",cocpt,"nombres figurés","nombre figuré")`).

1 La lemmatisation désigne un traitement lexical apporté à un texte en vue de son analyse. Ce traitement consiste à appliquer aux occurrences des lexèmes sujets à flexion (en français, verbes, substantifs, adjectifs) un codage renvoyant à leur entrée lexicale commune ("forme canonique" enregistrée dans les dictionnaires de la langue, le plus couramment), que l'on désigne sous le terme de *lemme* (Wikipédia).

2 https://fr.wikipedia.org/wiki/Racinisation

3 http://www.projet-ermitage.org/prof-expert.html

4 SWI-Prolog

A noter que ces items peuvent être exportés en format *php* afin d'être réinjectés dans la banque de données.

3.4. Tri et dénombrement

Les mot-clés (complétés manuellement par des termes mathématiques ne figurant pas forcément dans les mot-clés) sont stockés à part. Ils permettent d'extraire des différents champs ce qui concerne les mathématiques. La procédure de tri permet aussi de supprimer des éléments qui auraient échappé aux analyses antérieures.

Les occurrences de chaque mot pour chaque champ et chaque document sont dénombrées. Des tableaux de fréquences [document x mots] peuvent être produits pour les analyses ultérieures. Une ligne de la matrice donne le nombre d'occurrences de chaque mots d'un document. Ainsi les lignes représentent des documents et les colonnes les mots (l'ordre peut-être inversé). Des identifications de documents et les mots constituent les labels des lignes et des colonnes (ou l'inverse).

Les matrices étant souvent creuses (rares), une autre façon de préparer les données consiste tout d'abord de créer un dictionnaire des mots et un autre des documents puis de recenser la liste des triplets (index du document, index du mot, fréquence). D'autres structures peuvent être utilisées qui dépendent du logiciel considéré.

A signaler encore qu'une procédure de pré-traitement *tf-idf* (voir annexe 2) fait également partie de l'arsenal des outils testés dont deux usages sont possibles : remplacer le nombre d'occurrences d'un terme par les valeurs relatives (prétraitement de type 1) ou alors supprimer les termes de la matrice des occurrences dont les valeurs relatives sont trop faible (prétraitement de type 2). Il est également possible de conjuguer les deux usages (prétraitement de type 3).

A noter que le prétraitement de type 2 doit être appliqué avec précaution. En effet, en effectuant un prétraitement pourtant raisonnable de ce type sur la matrice [documents x tâches] de dimension 360x3679, la matrice obtenue, de dimension 360x281, ne contient presque plus aucun terme mathématique.

Le corpus utilisé dans ce travail est constitué de 468 documents. Le dictionnaire (l'ensemble des termes tous documents confondus) contient 718 mots. N'ont été retenus que les termes mathématiques et les verbes. La répartition des mots dans les documents est donnée dans le tableau 1 et le nombre de documents par mot dans le tableau 2.

	Nombre de mots en moyenne par doc.	Nombre de mots min[1]	Nombre de mots max
Avec multiplicité	78	9	380
Sans répétition	35	6	71

Tableau 1. Nombre de mots par document

	Nombre moyen de documents par mot	Nombre min de documents	Nombre max de documents
Avec multiplicité	51	1	993
Sans répétition	23	1	369

Tableau 2. Nombre de documents par mot

1 Les petites valeurs (le problème avec 6 mots uniques est 02.II.05, code gp37) sont présente dans les problèmes des premiers rallyes dont l'analyse de la tâche est absente.

4. Mise en oeuvre de différentes analyses

4.1. Introduction

Les méthodes ou modèles classiques de classification des données sont les analyses en clusters pour le cas purement métrique et les analyses factorielles (dont l'ACP est le cas prototypique) vectorielles. Par essence, ce sont des modèles déterministes. Sous l'influence du « machine learning », l'usage et la mise au point de modèles probabilistes ont explosé. Cette évolution est marquée par les deux éditions d'un ouvrage de référence en classification automatique : *Pattern classification* de Duda, Hart et Stork. Par rapport à la première édition publiée en 1973, l'édition de 2001 présente des nouveaux sujet tels que : *Neural networks*, *Statistical pattern recognition*, *Theory of machine learning*. Deux ouvrages récents dans le même domaine seront aussi fréquemment consultés : Bishop (2006), Murphy (2012).

Outre la classification, le modèle de données obtenu (par apprentissage non supervisé pour se plier au vocabulaire actuel) devra aussi pouvoir être utilisé pour classer de nouveaux documents et permettre une restitution fidèles et ainsi faciliter la restitution de données (*information retrieval*).

Les recherches actuelles se détournent des prédictions faites à l'aide de modèles déterministes (via par exemple la projection de nouvelles données dans l'espace des facteurs) et tendent à les remplacer par des modèles qui permettent de relativiser la pertinence des résultats obtenus.

L'exemple du polynôme d'interpolation proposé par Bishop (2006) que l'on résume en annexe (annexe XX1) montre bien cette façon de faire.

La littérature fourmille de terminologie encore peu stabilisée. En gros les méthodes de classifications sont de type génératif ou discriminant. En prenant X variable observable et Y variable cible, le modèle génératif recherche une loi probabilité sur X x Y. Dans le deuxième cas, il s'agit d'une une loi de probabilité conditionnelle $Y|X$.

De façon plus spécifique, Nie (s.d.) par exemple, restreint le modèle génératif au cas où les données sont (pourraient être) générées par un processus random avec des paramètres qu'un « apprentissage » ajuste aux données. Un tel modèle devrait permettre de traiter (classifier, prédire, inférer) de nouvelles données.

Un exemple d'apprentissage, parmi les plus simples, est donné par l'algorithme EM (Espérance-Maximisation, en anglais *expectation-maximization algorithm*). Cet algorithme itératif permet d'ajuster des paramètres à partir du calcul de coefficients de vraisemblance dont on cherche un maximum (voir annexe 4).

4.2. Analyses en clusters

Les analyses en clusters classiques[1] sont parmi les méthodes de classification les plus simples. Elles présentent toutefois l'inconvénient de rendre difficile la mise en évidence des éléments caractérisant les groupes. C'est un problème que nous avions rencontré lors d'un regroupement des revues en fonctions des thèmes traités (Berney & Pochon, 2000). Et nous imaginions alors une analyse en doubles clusters[2].

A l'époque, peu de travaux étaient disponibles hormis l'algorithme Bimax de Hartigan (1972 cité par Prelic & al, 2006) et présenté comme une méthode de référence.

1 Pour un survol des algorithmes à disposition on peut consulter Fahad & al (2014) qui fait la classification d'une vingtaine d'algorithmes avec quelques critères d'évaluation.

2 http://www.projet-ermitage.org/thema/stat-clu.htm ; http://www.projet-ermitage.org/thema/stat-dbl.htm

Depuis 2000, les techniques d'analyse en bi-clusters (ou co-clusters) se sont développées, notamment sous l'impulsion des analyses génétiques dans l'étude de l'expression des gènes.

Globalement, il s'agit de regrouper des informations selon deux dimensions et créer ainsi des blocs de données (les clusters). De nombreuses variantes existent. Les blocs peuvent être disjoints ou non, posséder une certaines structure ou non, avec superposition des attributs ou non. Le procédé peut être exhaustif (c'est-à-dire que la réunion des clusters recouvre l'ensemble des données) ou non ; il peut s'agir par exemple de trouver le maximum de blocs possédant une certaine structure.

Ces techniques peuvent s'avérer utiles pour mettre en relations diverses classifications (contrôle de la redondance)[1], mais moins pour définir des critères susceptibles d'être étendus à des documents extérieurs (dont les requêtes) au corpus de base. De plus, les classifications sont trop strictes. Toutefois, les techniques de « clustering flou » (*fuzzy* ou *soft clustering*) où chaque élément peut appartenir à plusieurs clusters seraient à expérimenter. On ne peut pas opposer totalement l'analyse en facteurs (*aspect analysis*) et l'analyse en clusters comme le détaille Hofmann (1999).

4.3. Modèles à variables latentes

Les modèles statistiques à variables cachées, latentes, sont issus des travaux des psychologues au début du XXe siècle dans l'analyse des résultats à des tests[2]. Il s'agissait alors de repérer les différentes composantes de l'intelligence, les « vecteurs de l'esprit »[3]. Les méthodes se sont diversifiées pour s'appliquer dans tous les domaines de la recherche scientifique à différents types de données (variables entières, nominales, etc.) et principalement consacrées à la réduction de grands tableaux de données. Ces variables cachées ont reçu le nom de facteurs, d'où l'appellation générique d'analyse factorielle (*factor analysis* en anglais).

Ces modèles d'analyse ont émigré dans le domaine de l'information textuelle. Leur usage part de l'hypothèse que les contenus des documents d'un certain corpus sont liés par quelques variables cachées, latentes. Dans ce cas, ces variables reçoivent le nom de thèmes, sujets ou *topics* en anglais, terme que nous utiliserons vu son apparition fréquente dans les tableaux ou autre documents délivrés par les algorithmes. Ces modèles font deux hypothèses[4] :

* Chaque document est décrit par un mélange (pondéré ou non, délimité ou non, unique ou non) de *topics*.

* Chaque *topic* est défini par une collection (pondérée ou non, délimitée ou non) de mots. Selon la méthode, deux *topics* différents peuvent ou ne peuvent pas être associés au même mot.

Brièvement dit, les *topics* d'un document le résume et les *topics* sont définis par des agglomérats de mot-clés[5].

Le but des analyses est alors de « découvrir » les variables latentes que sont les *topics*[6].

Il faut distinguer les analyses de type déterministe où les pondérations sont des composantes (vectorielles) dans des espaces de mots et les analyses de type probabiliste où les pondérations sont des probabilités (probabilité qu'un *topic* décrive un documents, probabilité qu'un mot définisse un

1 La tentative de mettre en relation *topics* et familles de tâches sera brièvement évoquée et fera l'objet d'une autre chronique.

2 Cette modeste origine semble oubliée des travaux actuels de plus en plus sophistiqués.

3 Quelques bribes historiques sont données dans Pochon (2012:89) et Bishop (2006:561).

4 https://medium.com/nanonets/topic-modeling-with-lsa-psla-lda-and-lda2vec-555ff65b0b05

5 Nous les appelions des méta-concepts ou super-concepts (http://www.projet-ermitage.org/thema/stat-svd.htm).

6 On retrouve parfois sous-jacent l'esprit des débuts de l'analyse factorielle où les variables n'ont pas le statut de constructions mais de découvertes. Elles semblent avoir une existence ontologique.

topic)[1].

Hormis l'analyse factorielle classique que certains redécouvrent, les techniques les plus simples souvent évoquées qui modélisent un ensemble de documents par des *topics* sont LSA (Latent Semantic Analysis ; le sigle LSI - Latent Semantic Index est utilisé pour désigner les procédures d'extraction de données basée sur LSA), pLSA (probabilistic Latent Semantic Analysis ; on trouve également le sigle pLSI pour son usage dans l'extraction de données), LDA (Latent Dirichlet Allocation) avec plusieurs perfectionnement (Murphy, 2012) et d'autres techniques plus récentes que nous n'examinons pas ici[2].

Nous mènerons les analyses en proposant cinquante *topics*. Ce nombre est vraisemblablement trop élevé. Il a été choisi, un peu au hasard, parce que l'analyse LSA en proposait 60 et qu'il y a une septantaine de tâches principales définies manuellement. Le choix du nombre de *topics* est délicat. S'il y en a trop peu, chaque *topic* recouvre un domaine très large et s'il y en a trop, certains risquent de devenir artificiels en prenant des combinaisons de mots particulières que l'on ne peut plus interpréter.

Notation

Dans cette chronique les notations suivantes sont adoptées. M est le nombre de documents et V le nombre de mots différents dans l'ensemble du corpus (dictionnaire)

w_{ij} nombre de fois où le mot w_j se trouve dans le document d_i

$d_i, i=1\dots M$ le *i*-ème document du corpus ou le vecteur $(w_{ij})_{j=1\dots M}$

$w_j, j=1\dots V$ le *j*-ème mot du vocabulaire ou le vecteur $(w_{ij})_{i=1\dots V}$ (ou $i=1\dots N$ si on ne s'occupe que des mots d'un document)

$z_k, k=1\dots K$ le *k*-ième topic ou le vecteur $(\delta_{ki})_{i=1\dots K}$ indiquant que le *topic i* est associé au *k*-ième terme (token) du document courant lors du processus génératif (ou le *topic* associé au k-ième token d'un document)

N_d longueur du document *d* en nombres de mots (y compris les répétitions)

n_{tot} nombre de mots du corpus ($\sum_{i=1}^{M} N_{d_i}$)

$p(w|z)$ poids probabiliste du terme *w* dans la détermination du *topic z* ($\sum_{w\in V} p(w|z)=1$)

$p(z|d)$ poids probabiliste du *topic z* pour le document *d* ($\sum_{k=1}^{K} p(z_k|d)=1$)

4.3.1. Evaluation des résultats

L'évaluation des résultats obtenus par ce type d'analyse n'est pas une mince affaire. Outre l'efficacité, la performance des algorithmes à laquelle nous ne nous intéressons pas dans cette chronique, il faut distinguer la stabilité de l'analyse et sa « qualité » qui peut concerner la classification et la restitution.

1 La classification des différents types d'analyses n'est pas chose aisée. On peut aussi distinguer les méthodes discriminatives, qui se focalisent sur les données connues pour en extraire les variables cachées et les méthodes génératives qui se concentrent sur les paramètres qui permettent de mieux générer le corpus.

2 https://towardsdatascience.com/lda2vec-word-embeddings-in-topic-models-4ee3fc4b2843

Interprétation

A l'origine de ces techniques, en psychologie, une qualité de l'analyse se révélait principalement dans la facilité d'interpréter les facteurs qui, vu le type de travaux, se trouvaient en petits nombres.

Dans les analyses de grands corpus textuels l'interprétation des *topics* peut s'avérer plus ardue. Une approche mentionnée (*labeled LDA*) par Murphy (2012:953) exploite l'existence de *tags* qui identifient les documents. L'analyse force les *topics* à correspondre aux *tags*. Un regard plus précis serait à porter sur les détails de cette technique qui paraît étrange au premier abord dans la mesure où l'élaboration des *tags* est en lui-même une tâche de classification. Gaussier & Yvon (2011:182) détaillent un peu la procédure de même que Ducrocq (2014).

La possibilité d'interpréter les *topics* en petit nombre, en s'aidant d'outils de visualisation reste un gage de bonne classification. Nous expérimenterons le système *LDAvis*[1] développé pour R (Sievert & Shirley, 2014) et porté sur Python (bmabey, 2018 ; Kapadia, 2019). Une méthodologie liée au système *Termite* est également proposée (Chuang, Manning & Heer, 2012).

Plusieurs auteurs suggèrent de représenter les *topics* par des nuages de mots (Tang, 2019) qui est une technique générale souvent proposée pour des textes de dimension raisonnable (Rul, 2019)[2].

Stabilité

La stabilité concerne principalement les analyses dont les algorithmes font appel à des processus itératifs ou d'échantillonnage. C'est le cas pour la plupart des modèles probabilistes. Il s'agit dans ces cas de maximiser un coefficient rendant compte de la structure recherchée lorsqu'il n'existe pas de solution analytique ou alors que la quantité des données empêchent de les mettre en oeuvre. Les résultats obtenus par ces algorithmes qui procèdent par approximations successives peuvent dépendre des valeurs initiales. Par exemple, la recherche d'un maximum global peut être obstruée par l'atteinte d'un maximum local. Dans cette chronique nous nous bornerons à la persistance des résultats lors de quelques répétitions de l'analyse, à des comparaisons entre des algorithmes différents et à l'influence du pré-traitement *tf-idf*. Dans la littérature, des coefficients de corrélation où des tables de réorganisation peuvent être utilisées (Mazarura & Waal, 2016) ou encore des distances (par exemple la distance de *Kullback Liebler*) entre des distributions (Steyvers & Griffiths, 2007). Cela concerne principalement la classification et souvent par voie de conséquence la restitution.

La robustesse (terme qui évoque les statistiques robustes) d'un algorithme peut être liée à la stabilité.

Qualité

La question de la « qualité » intrinsèque (nous utilisons ce terme général pour ne pas empiéter sur des termes qui recouvrent des analyses précises) est plus délicate. C'est en partie une question subjective. Il s'agit aussi de distinguer les deux aspects en partie liés : la classification et la restitution.

On peut tout d'abord considérer des solutions pragmatiques : par sondage des documents proches d'un document donné (pour classification) ou de documents proches de documents hors corpus (dont des requêtes) classifiés a posteriori (pour la restitution).

Une mise en relation avec une classification déjà effectuée par ailleurs constitue également une

1 Une introduction existe sous forme d'une vidéo : https://www.youtube.com/watch?v=IksL96ls4o0

2 Voir aussi http://www.sthda.com/french/wiki/text-mining-et-nuage-de-mots-avec-le-logiciel-r-5-etapes-simples-a-savoir

démarche évaluative. Il en sera brièvement question dans cette chronique et fera l'objet prochaine chronique.

La possibilité d'interpréter les *topics* en vue d'une classification est aussi un gage de qualité. Ce n'est par contre pas toujours possible, voire nécessaire, si l'analyse est menée dans le but de faciliter les restitutions.

L'usage et la mise en place d'un ajustement en continu, d'une mise au point graduelle (voir la partie concernant ce sujet), est aussi une réponse à cette difficulté de définir la qualité du point de vue de la restitution.

Dans le cas probabiliste des indices délivrent des informations sur la qualité des analyses. Ils semblent surtout utilisés à titre comparatif et servent par exemple à déterminer le nombre de *topics* le plus adéquat. Nous avons calculé quelques-uns de ces indices sans vraiment pouvoir les interpréter. Ils sont engrangés à titre de comparaison pour nos travaux ultérieurs.

Les deux exemples de coefficients fréquemment rencontrés dans la littérature sont la « *perplexity* » et la « *coherence* »[1]. Ils peuvent s'appliquer à chaque *topic* ou à l'ensemble du modèle.

Perplexité

La *perplexité* est une notion appartenant à la théorie de l'information. Elle est liée à l'entropie informationnelle. De façon formelle[2], la *perplexité* d'une distribution p est son entropie $H(p)$ élevée à la puissance 2 : $PP(p)=2^{H(p)}$. Elle situe l'entropie sur une échelle non logarithmique ce qui peut dans certains domaines mieux accorder la mesure donnée par l'entropie à l'intuition. Une connaissance parfaite (distribution centrée sur une moyenne, d'entropie quasi-nulle) correspond à la valeur 1 pour la *perplexité*. Une méconnaissance totale (distribution uniforme, entropie maximale $\log_2 n$ où n est le nombre des événements possibles) correspond à la *perplexité* maximum (n).

Elle est adaptée de façons diverses dans le but d'évaluer quantitativement les décompositions en facteurs latents.

Hofmann (1999) utilise cette notion pour comparer la performance prédictive de LSA et pLSA. Ce qui l'engage à probabiliser LSA, entreprise assez périlleuses. Dans son article, il n'y a pas d'information précise sur le mode de calcul de l'indice autre que l'indication que la *perplexité* se réfère à la moyenne logarithmique de l'inverse des probabilités (c'est-à-dire l'entropie) sur les données hors corpus.

L'article de Blei & al (2003) précise que l'indice de *perplexité*, utilisé de façon standard en modélisation du langage est décroissant avec l'augmentation du coefficient de vraisemblance sur les données de test. Il est équivalent à l'inverse de la moyenne géométrique de la vraisemblance par mot. Un score bas correspond à une meilleure performance de généralisation. Sans autre détail, la formule proposée pour déterminer la *perplexité* (*PP*) d'un ensemble de documents de test est (avec notre notation) :

$$PP(Dtest)=\exp\left(-\frac{\sum_{i=1}^{M}\log p(d_i)}{n_{tot}}\right)$$

Une transformation simple conduit à : $PP(Dtest)=\prod_{i=1}^{M}\left(\frac{1}{p(d_i)}\right)^{1/n_{tot}}$, formule qui correspond

1 Nous n'avons pas trouvé de littérature en français utilisant ces notions. Nous nous permettrons toutefois de franciser ces deux termes très suggestifs (perplexité et cohérence).

2 https://en.wikipedia.org/wiki/Perplexity

plus ou moins à l'énoncé textuel. Si l'on veut faire coïncider la formule et la description, ne faut-il pas admettre l'hypothèse d'indépendance des mots dans un document ?

Cette formule est également proposée par Griffith & Steyvers (2004) avec la remarque que la perplexité indique l'incertitude de prédire un mot, et que la valeur optimale correspond à la taille du vocabulaire.

Comme elle évolue en fonction du nombre de *topics,* les mêmes auteurs (Steyvers & Griffiths, 2007) utilisent la *perplexité* pour la recherche du nombre adéquat de *topics*.

Rosen-Zvi & al. (2004) introduisent une généralisation du modèle LDA en ajoutant une information sur les auteurs et procèdent à une évaluation de leur modèle utilisant la *perplexité* sur un ensemble de mots test. Ils remarquent que les performances de généralisation sont d'autant meilleures que la *perplexité* est faible sur un document choisi.

On trouve dans l'ouvrage de Murphy (2012:953) des informations sur l'origine de cette notion utilisée en linguistique computationnelle comme mesure de la qualité d'un modèle de langage naturel. Plus précisément pour évaluer la capacité de généralisation d'un modèle de texte à travers des sous-ensembles de documents. Elle fait intervenir l'entropie croisée entre deux distributions.

Avec plus ou moins de réticence, il considère que le modèle délivré par LDA est un modèle de langage, c'est-à-dire une distribution probabiliste sur des séquences de mots. Ce n'est évidemment pas un très bon modèle de langage puisque l'analyse ignore l'ordre des mots et considère les mots séparément, mais selon l'auteur cité, il peut être intéressant de comparer LDA à d'autres modèles grâce à cet indice.

A ce stade, nous retenons que c'est surtout une information relative, entre théorie et bricolage, qui permet des comparaisons dont les limites sont reconnues.

Comme les deux packages que nous utilisons, *topicmodels* (Grün & Hornik, 2011, 2018) et *text2vec* (Selivanov, Bickel & Wang, 2020), permettent de calculer cet indice[1], il nous a semblé utile de consulter le code de la fonction *perplexit*y de *text2vec* . Nous en déduisons la formule.

$$\exp\left(-\sum_{ij}\frac{w_{ij}}{n_{tot}}\log\left(\epsilon+\sum_{k}p(z_k|d_i)\,p(w_j|z_k)\right)\right)$$

qui peut s'écrire : $\exp\left(-\sum_{ij} freq(w_j\, dans\, d_i)\log\left(p(w_j|d_i)+\epsilon\right)\right)$

Cette formule est intéressante à plus d'un titre. Dans sa forme, elle correspond bien aux descriptions entropiques de la *perplexité*. De fait, elle correspond à l'entropie croisée citée par Murphy. A noter qu'à une constante additive près, valable pour tout un corpus, elle vaut la divergence de *Kullback-Leibler*. Elle donne la distance entre la distribution effective des mots dans les documents et la distribution calculée dans le modèle. Elle a donc une légitimité hors du modèle exporté de la linguistique.

Cohérence

La *cohérence* est un autre indice, plus adapté pour l'évaluation d'un modèle selon Kapadia (2019) qui expose une méthode de recherche du nombre de *topics* qui maximise cet indice. La *cohérence* peut concerner chaque *topic* séparément ou l'ensemble du modèle. Il y a plusieurs mesures possibles.

Stevens & al (2012) précise que la *cohérence* d'un *topic* est une mesure du degré de similarité

1 https://www.rdocumentation.org/packages/text2vec/versions/0.6/topics/perplexity ;
https://rdrr.io/cran/text2vec/man/perplexity.html ;
https://www.rdocumentation.org/packages/topicmodels/versions/0.2-9/topics/perplexity

sémantique entre les termes les plus représentatifs d'un *topic* (de poids élevés). Cette mesure devrait permettre de distinguer les *topics* qui ont une interprétation sémantique de ceux qui émergent de l'outillage statistique. Ils proposent diverses méthodologies pouvant s'appuyer sur des juges ou par une automatisation reposant sur les mots du corpus ou ceux d'un corpus externe.

Dans les procédures automatiques, plus il y a de mots définissant un *topic* présents en même temps dans un ensemble de documents de référence (le corpus traité ou un corpus extérieur) et plus la *cohérence* est élevée. Les définitions peuvent légérement varier selon les auteurs.

$$coherence(T) = \sum_{u_i v_j \in T} score(v_i, v_j, \epsilon)$$

où T est l'ensemble des mots décrivant un *topic* (où une partie des mots, ceux de poids élevés) et ϵ (souvent 1) comme facteur de lissage.

Différentes mesures peuvent être utilisées pour définir le score par exemple :

- métrique UCI : $\log \frac{p(vi, vj)+\epsilon}{p(v_i)\,p(v_j)}$ où $p(x, y)$) est la fréquence de co-occurence des termes *x* et *y* dans un corpus extérieur (par exemple : Wikipedia) et $p(x)$ la fréquence du terme *x* dans le même corpus ;

- métrique UMass : $\log \frac{D(v_i, v_j)+\epsilon}{D(v_j)}$ avec *D*(*x*,*y*) nombre de documents (du modèle) contenant *x* et *y* et *D*(*x*) nombre de documents (du modèle) contenant *x*.

Pour comparer les modèles globalement, il s'agit d'agréger les cohérences individuelles des *topics*. Deux méthodes sont proposées : (1) la cohérence moyenne ou (2) l'entropie de la cohérence pour tous les *topics*. La méthode 1 fournit un résumé rapide de la qualité alors que l'autre fournit une alternative qui permet de distinguer deux situations intéressantes. En effet, puisque l'entropie mesure la complexité d'une distribution de probabilité, elle peut différencier les distributions uniformes des multimodales. Cette distinction est utile lorsque l'utilisateur préfère une qualité uniforme plutôt qu'un saut entre *topic* de haute ou de basse qualité. Les auteurs précisent que l'on calcule l'entropie en laissant tomber le *log* et ϵ de chaque fonction score.

Une hypothèse est qu'une relation existe entre cohérence et qualité de la classification. Les auteurs cités suspectent qu'une haute cohérence des *topics* et du modèles vont fournir de meilleures classifications.

Le package *text2vec* propose une fonction pour le calcul de la *cohérence* selon plusieurs mesures. Il se réfère à l'article de Röder & al (2015). La cohérence basée sur l'information mutuelle (PMI) (métrique UCI) est celle qui se rapproche le plus des scores humains.

Tang (2019) propose un guide du débutant pour l'utilisation du modèle LDA. Le nombre de *topics* adéquat est défini par la *cohérence* sans précision sur le coefficient utilisé. Jones (2019) fait de même en utilisant le package *textmineR*. Quant à Kumar (2918), il propose une marche à suivre, liée à LDA, pour le calcul de la *cohérence* en utilisant un package Python et la mesure C_v en distinguant bien mesure intrinsèque et extrinsèque.

En définitive, il n'est pas aisé d'estimer la qualité d'une décomposition en variables latentes. La plupart des indices sont de types comparatifs. Ils servent souvent à déterminer le nombre adéquat de *topics*.

Mentionnons encore deux coefficients utilisés dans les représentations graphiques : la *pertinence* et la *saillance* (*saliency*).

Pertinence et Saillance

Sievert & Shirley (2014) introduisent un coefficient, la *pertinence*, qui permet de façon flexible d'ordonner les termes définissant un *topic*. Ce coefficient rend compte de la pertinence du terme dans la définition du *topic*.

Pour un *topic* donné *z*, la *pertinence* d'un terme *w* est défini par[1] :

$$r(w, z|lambda) = \lambda \log p(w|z) + (1-\lambda) \log \frac{p(w|z)}{p(w)}$$

où λ est compris entre 0 et 1, $p(w|z)$ est la probabilité de *w* pour *z* (le poids de *w* dans *z*). $p(w)$ dénote la probabilité d'apparition du terme dans le corpus.

Les auteurs présentent une étude concernant la recherche d'une valeur de λ optimale.

Chuang & al (2012) proposent tout d'abord de définir la « *distinctiveness* » (*distinguabilité*) d'un terme *w* comme la divergence de *Kullback-Leibler* entre la vraisemblance que le mot observé soit généré par le *topic* z : $p(z|w)$, et la probabilité marginale $p(z)$, vraisemblance qu'un mot sélectionné au hasard soit généré par le *topic* z :

$$distinctiveness(w) = \sum_z p(z|w) \log \frac{p(z|w)}{p(z)}$$

Cette formulation décrit (dans le sens de la théorie de l'information) combien est informatif un terme spécifique *w* pour définir les *topics* par rapport à un terme *w*' sélectionné au hasard. Si par exemple un mot apparaît dans tous les *topics*, il nous apporte peu sur le mélange des *topics* des documents. En conséquence son score de *distinguabilité* sera faible.

La *saillance* d'un terme *w* est définie par : $saliency(w) = freq(w) \times distinctiveness(w)$.

Proximité de documents : calcul direct

Retour à nos données. Au vu de l'état d'avancement de notre réflexion. l'usage de ces indices est prématuré. Nous en calculerons quelques-uns en vue d'un usage ultérieur. Pour notre évaluation, nous comparerons les réponses (restitutions) basées sur la distance entre documents calculée sur les données brutes à celles basées sur la distance calculée sur les variables latentes. Cet élément de comparaison va se baser sur un document du corpus, trois documents de la banque de problèmes hors corpus et deux documents-requêtes. Nous espérons en tirer, en testant également une technique de visualisation, quelques conclusions aussi bien du point de vue de la classification que de la récupération.

Le document du corpus est le 10e (3d18). Les trois documents corpus sont les problèmes de codes : ud104, ud85, ud87. Les deux requêtes sont "*rectangle périmètre aire quadrillage*" (req1) et "*dénombrer combinatoire ordre déduire*" (req2)[2]. Les cinq documents les plus proches de ces documents test (selon la distance du cos ; plus la valeur s'approche de 1 et plus les documents sont proches) sont donnés dans le tableau 3 sans et avec prétraitement *tf-idf*.

Pour chaque analyse, lorsque c'est possible, on comparera ainsi les analyses sur les données sans et avec prétraitement *tf-idf*. La comparaison inter-méthode se fera ultérieurement sur la base des onze documents les plus proches.

1 La formule utilisée dans le système LDAvis pour classer les termes, la formule de la pertinence omet le logarithme.

2 Il se trouve que dénombrer ne fait pas partie des mots du corpus. C'est une négligence qui sert à vérifier que programmes ne sont pas bloqués dans ce cas.

Document	Sans tf-idf	Avec tf-idf
3d18 (corpus)	3d18 lr19 gp126 lr21 3d25 1.00 0.43 0.41 0.39 0.38	3d18 op27 pr20 gp34 gp37 1.00 0.21 0.21 0.20 0.16
ud104	gp115 gp26 gp69 gm6 gp25 0.63 0.60 0.58 0.56 0.52	gp115 gp48 gm6 gp26 gm5 0.52 0.45 0.44 0.42 0.41
ud85	nu15 nu18 nu20 nu27 nu21 0.71 0.69 0.69 0.65 0.62	nu15 nu18 nu20 nu27 op93 0.79 0.68 0.66 0.64 0.62
ud87	op75 pr14 op46 op96 op86 0.45 0.44 0.43 0.43 0.41	op96 op86 op46 nu5 op47 0.45 0.44 0.44 0.39 0.39
req 1	gp3 gp8 gp131 gp16 gp103 0.71 0.67 0.61 0.61 0.60	gp3 gp8 gp16 gp131 gp129 0.74 0.63 0.60 0.59 0.54
req 2	lr23 lr18 lr11 op6 op62 0.35 0.32 0.24 0.22 0.20	lr23 op6 lr18 lr11 lr21 0.40 0.32 0.32 0.28 0.19

Tableau 3. Les cinq documents les plus proches de chaque document test (données brutes, distance du cos)

Commentaire

On notera une meilleure correspondance entre la distance calculée sans et avec transformation *tf-idf* pour les requêtes et pour le document ud85.

Il est difficile à ce stade d'estimer l'influence du prétraitement. Une analyse qualitative détaillée pourra être menée dans un deuxième temps. Remarquons de façon rapide pour :

- ud104 (Les rubans transparents) : la réponse gp115 (Les deux rectangles) n'est pas fameuse.

- ud85 (Le plus grand produit) : la réponse nu15 (Drôle de nombre) est assez bonne.

- ud87 (Menteur et menteur) : les réponses op75 (Concours de pêche) et op96 (Les prunes) sont bonnes bien qu'elles se basent sur des raisonnements numériques alors que la tâche ud87 est purement logique. On notera que la distance qui la sépare de la requête est l'une des plus grandes du lot.

- req1 : la réponse gp3 (Les carrés d'Alex et de François) est assez bonne.

- req2 : la réponse lr23 (Photos de footballeurs) n'est pas bonne. On notera que c'est la réponse la plus éloignée de la requête de tout le lot.

4.3.2. Modèles déterministes à variable latente

4.3.2.1. Analyse en composantes principales (ACP)

L'ACP (PCA) est une technique qui a une longue histoire. Les facteurs pourraient tout à fait jouer le rôle de ces thèmes recherchés. Toutefois trois problèmes se posent *a priori*. Le premier est d'ordre technique, le deuxième est relatif au modèle sous-jacent et le troisième relève de l'usage et de l'interprétation[1].

1 Il faut aussi noter que l'analyse factorielle classique traite de variables continues, ce qui dans le cas présent n'est pas

- Techniquement : les algorithmes habituels ne peuvent pas traiter de trop grandes matrices creuses. Toutefois, certains travaux semblent utiliser avec profit une méthode adaptative neuronale (GHA - Generalized Hebbian Algorithm) pour calculer un nombre réduit de nouvelles dimensions représentatives des données (Delichère & Memmi, 2002).

- Modèle sous-jacent : Les auteurs de l'étude précédente notent que PCA et GHA sont des méthodes purement linéaires[1] qui ne peuvent capturer que des corrélations linéaires entre les données. Ils proposent d'envisager d'autres méthodes pour dépasser cette limitation comme l'analyse en composantes indépendantes ou les cartes de Kohonen (ils citent les travaux de Hérault & Jutten (1994), Ritter & Kohonen (1989), Kohonen (1998)).

- Usage et interprétation : Par ailleurs, il n'est pas évident dans ce contexte d'utilisation d'interpréter les notes en facteurs ou les *communalités* d'autant plus si elles sont négatives. De plus, « l'espace culturel » de l'ACP et le « roman » de son utilisation sont assez éloignées, du moins pour nous, de l'analyse de corpus de texte.

Nous n'avons pas exploré plus avant la voie proposée par Delichère & Memmi (2002), ni examiné les variantes proposées. Nous aurions pu aussi nous intéresser aux extensions probabilistes de la PCA (Bishop, 2006 ; Murphy, 2012 ; Chen, Du & Vuyyuru, 2018).

Mais nous étions davantage tentés par l'exploration des modèles à variables latentes (LSA-pLSA-LDA) que nous avions croisés lors de travaux antérieurs et qui ont proliféré ces dernières années sous l'impulsion du « *machine learning* ». Sans compter que de nombreux packages existent qui les mettent en oeuvre. Leur « roman interprétatif » est aussi plus en adéquation avec le domaine d'application bien que du point de vue technique des rapprochements entre toutes les méthodes sont possibles.

4.3.2.2. LSA

LSA, Latent Semantic Analysis, est la plus simple et la plus ancienne des analyses qui mettent en évidence des traits latents à une famille de documents. Son usage principal est l'indexation de corpus de documents (on distingue parfois l'analyse LSA de son usage LSI : Latent Semantic Indexing). Elle a fait l'objet de plusieurs propositions méthodologiques (Gefen & ali, 2017). L'idée centrale est de décomposer la matrice [documents x mots] en une composante [documents x *topics*] et une composante [mots x *topics*] en utilisant simplement la diagonalisation SVD. C'est évidemment une méthode déterministe que Murphy (2012:947) déclare équivalente à PCA sans plus de précision. Nie (s.d.) parle de très semblable (*closely relate*d) et Obozinski (2012) de presque identique.

Les calculs sont examinés de plus près dans l'annexe 3.

Mise en oeuvre

Toutes les analyses latentes vont se baser sur le jeu de données (lemmatisées en *Prolo*g : noms et adjectifs au singulier, verbes à l'infinitif) constitué de 468 fiches et des 720 mots mathématiques et verbes utilisés dans les résumés, l'analyse de tâches, les consignes et les concepts. Deux mots qui apparaissent très fréquemment 'être' et 'nombre' sont encore supprimés du vocabulaire.

Les données initiales subissent également un prétraitement de type 1 pour obtenir un deuxième jeu de données de même dimension.

---

un handicap puisque les résultats attendus ne sont pas des valeurs discrètes. L'analyse en composante principale serait même plus adaptée qu'une analyse factorielle discrète comme l'analyse des correspondances multiples (MCA) (https://fr.wikipedia.org/wiki/Analyse_des_correspondances_multiples).

1 C'est aussi le cas de LSA.

Nous utilisons le package R, *lsa* (Wild, 2015) pour réaliser l'analyse avec cinquante *topics* (en mode automatique le système propose une soixantaine de *topics*) (de fait nous aurions pu directement utiliser la fonction *svd* fournie par R). Il est possible de vérifier que les deux matrices U et V sont orthogonales.

Le tableau 4 présentent à titre d'exemple quelques *topics* (analyse sans transformation *tf-idf*) avec les cinq mots dont les composantes sont positives maximales.

Topic 10	aire partir avoir minute calculer 0.59 0.24 0.11 0.10 0.09
Topic 11	figure aire euro cercle vendre 0.53 0.30 0.19 0.17 0.16
Topic 12	face aire base point droite 0.59 0.29 0.23 0.19 0.14
Topic 13	heure minute durée temps figure 0.52 0.46 0.21 0.18 0.17

Tableau 4. Quelques *topics* (sans *tf-idf*)

Topic 10	gm19 gp120 gp121 gm17 gm22 0.22 0.18 0.17 0.17 0.16
Topic 11	Al17 gp111 gm19 gp113 fn14 0.23 0.19 0.19 0.18 0.17
Topic 12	3d51 3d69 al20 3d60 lr25 0.19 0.19 0.18 0.16 0.16
Topic 13	al10 op71 gm23 gm12 pr28 0.43 0.25 0.27 0.22 0.21

Tableau 5. Quelques documents et les *topics* principaux associés (cas sans *tf-idf*)

On en déduit que le *topic* 10 concerne avant tout les problème de mesure, le *topic* 11 plutôt de géométrie plane, le 12 de géométrie 3d et le *topic* 13 des problèmes faisant intervenir le temps.

Si l'on regarde les *topics* associés aux documents (tableau 5), les résultats ne sont pas contradictoires. On constate que si un mot se détache nettement pour chaque *topic*, les *topics* sont plus également distribués entre les différents documents.

Le package R *quanteda.textmodels* (Benoit, Watanabe & al, 2013) que nous avons découvert ultérieurement (à travers l'exemple proposé avec l'ancien package *quanteda* par Wang (s.d.)) propose quant à lui une fonction pour appliquer un modèle à de nouvelles données. Nous l'utiliserons à titre de comparaison.

Les données de base à utiliser sont fournies par la matrice [document x mots]. Le chemin est bien balisé pour adopter une mise en forme matrice « rare » (ou creuse). L'algorithme délivre un objet contenant les trois matrices habituelles et leur produit. Les valeurs obtenues sont les mêmes que dans l'analyse précédente. Par contre, les composantes des documents hors corpus présentent quelques variations minimes qui sont répercutées sur la proximité des documents ainsi que le montre le Tableau 6 (quatrième colonne). Les raisons de ces différences seront à examiner ultérieurement. Cette analyse sera répertoriée par LSA Q[1]. Les résultats obtenus avec cette version ne seront plus utilisés dans les comparaisons ultérieures.

Proximité de documents via variables latentes

Le tableau 6 présente les 5 documents les plus proches des documents test avec la distance du cosinus sur les composantes dans l'espace latent. On trouve tous d'abord les résultats obtenus directement avec *SVD*, sans et avec transformation *tf-idf* puis à l'aide du package *quanteda.*

1 Ces tableaux peuvent paraître rébarbatifs. L'information devrait pouvoir servir à soumettre les résultats à une expertise humaine. Le lecteur intéressé a toujours le loisir de se rapporter à la banque de problème http://www.projet-ermitage.org/ARMT/

Document	Sans transformation tf-idf	Avec tf-idf	LSA Q (dans tf-idf)
3d18 (corpus)	3d18 lr19 al4 op80 op24 1.00 0.77 0.74 0.73 0.73	3d18 pr20 lr6 pr33 op2 1.00 0.76 0.76 0.67 0.63	3d18 lr19 op80 al4 nu24 1.00 0.70 0.64 0.57 0.54
ud104	gp115 gp26 gp69 gm6 gp25 0.627 0.601 0.578 0.562 0.524	gp115 gp48 gm6 gp26 gm5 0.524 0.450 0.438 0.419 0.413	gp26 gp69 gp115 lr7 gm5 0.731 0.69 0.62 0.57 0.54
ud85	nu15 nu18 nu20 nu27 nu21 0.709 0.687 0.685 0.650 0.616	nu15 nu18 nu20 nu27 op93 0.789 0.678 0.663 0.640 0.623	nu21 nu18 nu15 nu20 nu27 0.69 0.67 0.67 0.66 0.62
ud87	op75 pr14 op46 op96 op86 0.452 0.441 0.430 0.427 0.414	op96 op86 op46 nu5 op47 0.450 0.442 0.439 0.387 0.385	op96 op75 op46 pr14 op47 0.79 0.78 0.72 0.71 0.69
Req 1	gp8 gp131 gp3 gp129 gp16 0.821 0.806 0.800 0.799 0.782	gp3 gp108 gp8 gp64 gp65 0.917 0.902 0.901 0.837 0.830	gp131 gp129 gp8 gp16 gp3 0.74 0.65 0.60 0.60 0.52
Req 2	lr18 op32 nu10 op62 pr37 0.541 0.536 0.484 0.473 0.472	lr18 lr1 lr23 op21 lr5 0.89 0.82 0.81 0.79 0.78	lr18 op32 op52 gp71 pr37 0.48 0.43 0.42 0.41 0.41

Tableau 6. Proximité de documents via variables latentes (selon *topics* LSA, distance du cos)

Commentaire

A ce stade on ne compare que les résultats obtenus sans et avec transformation *tf-idf*. La comparaison inter analyse se fera ultérieurement.

3d18 : C'est le document qui fait office de contrôle. Les différences faibles entre les distances peut expliquer la différence entre les traitements avec ou sans *tf-idf*.

ud104, ud85 et ud87 : Pour ces documents, il s'agit tout d'abord de chercher les vecteurs correspondant en portant sur la liste des mots initiaux (le dictionnaire) les fréquences des mots correspondants du nouveau document. Globalement les résultats se recoupent. On notera le bonne correspondance pour le document ud85.

Document-requêtes : En ce qui concerne la première requête ("rectangle", "périmètre", "aire", "quadrillage") les résultats ne sont pas identiques bien qu'il y ait des recoupements. On note aussi le peu de différence entre les distances.

Les résultats pour la deuxième requête ("dénombrer","combinatoire","ordre", "déduire") sont assez similaires. On notera que le terme 'dénombrer' n'ayant pas été repéré comme terme mathématique la recherche ne se fait que sur trois mots.

Globalement les réponses sont assez bonnes. De façon plus précise :

- ud104 (Les rubans transparents) : la réponse gp26 (La bannière du château) est meilleure que gp115 (Les deux rectangles).

- ud85 (Le plus grand produit) : la réponse nu15 (Drôle de nombre) est meilleure que nu21 (Le ruban des nombres).

- ud87 (Menteur et menteur) : les réponses op75 (Concours de pêche) et op96 (Les prunes) sont bonnes bien qu'elles font appel à des raisonnements numériques alors que la tâche proposée par

ud87 purement logique.

- req1 : les restitutions gp3 (Les carrés d'Alex et de François) et gp8 (La boîte) sont meilleures que gp131 (Comparaison de figures) qui est toutefois aussi cohérente avec la requête.

- req2 : la transformation *tf-idf* accentue la proximité.

Conclusion intermédiaire

Ainsi, les résultats ne sont pas trop mauvais. Ils semblent plus incisifs avec la transformation *tf-idf*. Il y a recoupement entre les requêtes avec ou sans *tf-idf*.

On notera, sans pouvoir en tirer une conclusion que les sommes de répétitions absolues et relatives sont a peu près proportionnelles dans le cas de la première requête ce qui n'est pas les cas pour la seconde (Tableau 7).

	rectangle	périmètre	aire	quadrillage
Répétitions	706	141	521	156
Coeff. tf-idf	10.9	4.3	8.3	3.6

	dénombrer	combinatoire	ordre	déduire
Répétitions	0	18	79	266
Coeff. tf-idf	0	1	2.8	3.7

Tableau 7. Fréquences absolues et relatives (tf-idf) des mots présents dans les requêtes

Notons que nous n'avons pas normalisé selon *tf-idf* les documents hors corpus. Remplacer les fréquences absolues par des relative ne changerait rien aux résultats puisque la distance du cos est invariante par homothétie (voir aussi annexe 2 pour une petite subtilité).

La question de savoir l'influence de la longueur du document reste ouverte. La transformation *tf-idf* joue peut-être en partie un rôle de normalisation en amoindrissant la différence entre l'absence d'un mot et la présence de mots rares.

Quant aux composantes négatives, il est possible de les interpréter comme en analyse factorielle comme des facteurs niés. Leur sens reste toutefois obscur. De plus, lorsqu'elles sont toutes négatives, nous avons écarté le *topic* correspondant. C'est le cas du *topic* 1.

Dans leur article Coccaro & Jurafsky (1998) montrent que l'analyse LSA a un certain pouvoir prédictif sur les mots suivants dans un texte sémantiquement homogène. Ils en dérivent la probabilité de l'apparition d'un mot dans un « contexte ».

Une méthodologie très complète est proposée par Gefen, Endicott, Fresneda & al, (2017) qui resterait encore à exploiter.

Il est intéressant de noter qu'une ACP menée sur les *topics* répartit la variance de façon à peu près égale sur 50 facteurs.

4.3.3. Modèles probabilistes à variables latentes

Selon l'approche LSA l'information sémantique peut être déduite d'une matrice de co-occurrences mot-document et la réduction de la dimension fait partie du processus de mise en évidence de cette information.

Les modèles probabilistes admettent certainement ce point de vue. Par contre plutôt que d'exprimer cette information sous la forme de coordonnées pour les mots et les documents dans un espace euclidien, les propriétés sémantiques des mots et des documents sont exprimées en terme de probabilités par rapport à des thèmes. D'une représentation géométrique euclidienne on passe à une représentation de géométrie projective sur un simplexe (représentation triangulaire, tétraédrique, etc.). Un *topic* déterminé par trois mots de « coordonnées $p(w_i|z), i=1,2,3$ par exemple, peut se représenter comme le centre de gravité d'un triangle de sommets w_j chacun de poids $p(w_i|z)$) et plus généralement dans un simplexe de dimension *K*-1.

Le modèle probabiliste ne propose pas de « composantes » négatives dont l'interprétation n'est pas aisée dans LSA. A noter qu'il existe toute une classe de factorisations non négatives (dont NMF) qui pourraient être utilisées pour poursuivre ce travail dans un version déterministe. Shashanka, Raj & Smaragdi (2008) montrent qu'il y a correspondance entre les modèles à variable latentes et les méthodes de factorisation des matrices.

4.3.3.1. pLSA

PLSA, Probabilistic Latent Semantic Analysis, est une méthode basée sur un modèle probabiliste des classes latentes[1] proposée principalement par Hofmann (1999, 2017).

Son article, paru à plusieurs reprises, présente pLSA comme une variante probabiliste de LSA en définissant un modèle génératif des données. Le terme « *aspect model* » est utilisé mais peu repris dans la littérature (le terme anglais *aspect* est l'équivalent de facteur).

A chaque document d_i est associé la famille de variables latentes $(z_k)_{k=1\ldots K}$ selon une distribution discrète $p(z_k|d_i)$ (probabilité du *topic* z_k dans le document d_i) et chaque variable latente est définie par une famille de mots $(w_j)_{j=1\ldots N}$ également selon une distribution discrète $p(w_j|z_k)$ (probabilité de w_j sous le *topic* z_k).

La probabilité d'une co-occurence d'un document et d'un terme peut s'exprimer à l'aide de la probabilité des *topics* associés et de la probabilité qu'un mot soit associé au *topic*. Avec l'hypothèse d'indépendance des *topics* z_k et de $d_i \cdot w_j$ par rapport à z_k on a :

$$p(d_i, w_j) = p(d_i)\, p(w_j|d_i) \quad \text{et} \quad p(w_j|d_i) = \sum_k p(w_j|z_k)\, p(z_k|d_i)$$

Finalement (version asymétrique) : $p(d_i, w_j) = p(d_i) \sum_k p(w_j|z_k)\, p(z_k|d_i)$

ou aussi (version symétrique) : $p(d_i, w_j) = \sum_k p(z_k)\, p(d_i|z_k)\, p(w_j|z_k)$ (1)

Ensuite, on forme le coefficient de vraisemblance L (probabilité de trouver les données avec les paramètres) qu'il s'agit de maximiser :

$$L = \prod_{ij} p(d_i, w_j)^{n(d_i, w_j)}$$

où $n(d_i, w_j)$ est le nombre d'occurrences du mot w_j dans le document d_i . Cela revient à maximiser le log :

$$\boldsymbol{L} = \log(L) = \sum_{ij} n(d_i, w_j) \times \log p(d_i, w_j)$$

en utilisant l'algorithme EM (Expectation-Maximization) qui est un algorithme général pour traiter les mélanges de distributions (*mixture distribution*) (annexe 4).

La formule (1) peut s'écrire sous forme matricielle : $(p(d_i, w_j)) = U\, S\, V^t$ où :

1 https://en.wikipedia.org/wiki/Latent_class_model

- U de dimension M x K est formée des probabilités conditionnelles $p(d_i|z_k)$. Les sommes des colonnes (distribution du *topic* sur les documents) valent 1.

- V de dimension N x K est formée des probabilités conditionnelles $p(w_j|z_k)$. Les sommes des colonnes (distribution des mots sur le *topic*) valent 1.

- S diagonale de dimension K x K donne la fréquence-probabilité de chaque *topic* dans le corpus.

La matrice $(p(di,wj))$ est une version de co-occurence relative approchée de la matrice [documents x mots] originale A.

$$(p(di,wj)) \approx \frac{A}{\sum A} \quad .$$

Le modèle pLSA peut aussi s'exprimer comme un modèle génératif avec la distribution discrète p :

Pour chaque document d du corpus choisi selon $p(d)$:

- on choisit un *topic* z selon une distribution discrète associée à d ;

- on génère un terme w selon une distribution discrète associée à z.

Cela donne la formule conforme par ailleurs avec les lois des probabilité :

$p(w,d,z) = p(d)\,p(z|d)\,p(w|z)$ et donc

$p(w,d) = \sum_z p(w,d,z) = p(d) \sum_z p(z|d)\,p(w|z)$ (formule asymétrique)

Modèle graphique

Le modèle pLSA peut-être représenté graphiquement par les figures 1a et 1b. Les plages grisées correspondent aux ensembles des données connues (ensembles d'entraînement).

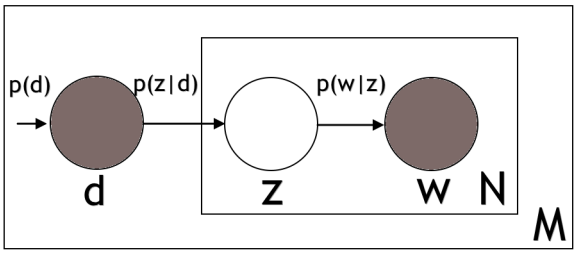

fig 1a. Modèle graphique de pLSA asymétrique

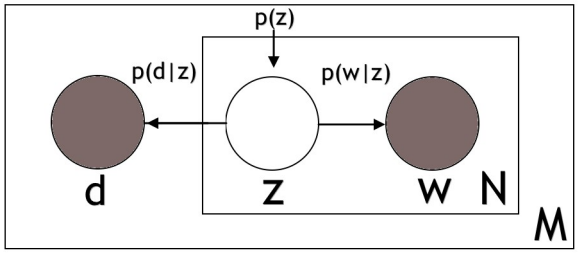

fig 1b. Modèle graphique de pLSA symétrique

Mise en oeuvre avec R[1]

Cette analyse à été menée avec les même données que précédemment en utilisant le package R *svs*

1 Analyse référée par la suite pLSA R

(Plevoets, 2016) qui fait référence à Hofmann (1999). Deux options sont possibles : sans symétrie ou avec. Avec symétrie ce sont les matrices U et V qui sont délivrées. On vérifie que les sommes des colonnes valent bien 1. On peut dans ce cas aussi vérifier que l'erreur moyenne entre la matrice de co-occurences relatives et $U S V^t$ vaut 3.8 x 10^{-23} avec de petites différence locales.

L'option sans symétrie délivre la matrice $(p(z_i|d_j))$ à la place de U ce qui est utile pour examiner le profil des documents. On a $p(z|d)=\frac{p(z)\,p(d|z)}{p(d)}=\frac{p(z)\,p(d|z)}{\sum_z p(z)\,p(d|z)}$.

Le tableau 8 présente quelques *topics* accompagnés des probabilités-fréquences des mots qui les définissent. Toutes les valeurs sont positives et leur somme sur l'ensemble des mots vaut 1.

Topic 1	triangle angle équilatéral figure pouvoir 0.413 0.058 0.037 0.035 0.030
Topic 2	chiffre écrire unité former premier 0.360 0.065 0.050 0.038 0.036
Topic 3	cube face arête former compter 0.438 0.113 0.046 0.027 0.022

Tableau 8. Quelques *topics* avec p(w|zi) (sans *tfi-df*)

Quant au tableau 9, il présente le profil des documents selon les *topics* (cas asymétrique). Les coefficients sont les probabilités-fréquences des *topics* constitutifs des documents. Toutes les valeurs sont positives et leur somme sur l'ensemble des mots vaut 1.

On constate que les *topics* 3 et 28 sont fortement liés aux problèmes de géométrie 3d.

Le tableau 10 correspond au cas symétrique. Il présente le profils des *topics* selon les documents.

3d1	3 34 29 14 10 0.649 0.066 0.048 0.048 0.042
3d10	45 28 6 25 38 0.231 0.125 0.121 0.103 0.102
3d11	28 3 1 6 42 0.577 0.241 0.045 0.026 0.025
3d12	28 1 34 41 9 0.332 0.182 0.170 0.131 0.054

Tableau 9. Quelques documents accompagnés de leur profil p(z|d) (sans t*fi-df*)

Topic 1	gp6 gm15 gm6 gp73 gp132 0.063 0.058 0.048 0.045 0.038
Topic 2	gp16 gp108 gp131 gp68 gp29 0.045 0.033 0.032 0.032 0.029
Topic 3	al17 al6 al11 al19 op77 0.203 0.124 0.084 0.073 0.062

Tableau 10. Quelques *topics* selon les document p(d|z) (sans *tfi-df*)

Proximité de documents via facteurs latents

Les articles concernant l'indexation sont assez avares d'information en matière de recherche de documents à partir de l'analyse. C'est lié au problème de l'intégration de documents hors corpus dans le format du modèle. Pour certains, c'est difficile (Nie, présentation sans date) ou Steyvers & Griffiths (2007)[1] ou disent qu'il ni a pas de voie naturelle pour assigner un poids en *topics* à un

1 « The pLSI model does not make any assumptions about how the mixture weights θ [c'est-à-dire les poids des topics pour chaque document] are generated, making it difficult to test the generalizability of the model to new

nouveau document (Blei & al, 2003) ou carrément impossible (Brooks, 2015). Pour Gaussier & Yvon (2011), plus nuancés, il n'existe pas de méthode fondée théoriquement pour affecter des paramètres à de nouveaux documents.

Certains proposent de trouver le *topic* le plus probable à partir d'un mot-clé puis le document le plus probable correspondant au *topic*.

Hofmann propose l'algorithme TEM (Tempered EM) pour évaluer le profil d'un document *q* hors corpus selon les *topics*

Pour le moment nous ne nous engageons pas dans cette voie qui ne semble jamais reprise ou commentée mis à part par Gaussier & Yvon (2011) qui le présente comme un palliatif pratique « théoriquement problématique ».

Chien & Wu (2008) proposent de traiter un document hors corpus avec la distance donnée par :

$$p(d_i|q)=\sum_{w_j\in q} p(d_i|w_j)\,p(w_j|q)=\frac{\sum_{w_j\in q} p(w_j|d_i)\,p(d_i)}{\sum_{d'} p(w_j|d')\,p(d')}\,p(w_j|q)$$

Cette méthode reprend les données « brutes » et ne présente pas un intérêt dans le cadre de l'utilisation des *topics*.

La tentation est grande de calculer directement $p(q|z)$ à partir des $p(w_i|z)$ avec w_i termes utilisés dans *q*. Le problème est que les mots ne sont pas indépendants. Mais on peut concevoir que cette information sur la dépendance existe bien dans le corpus de base.

Chien & Wu (2008) et Bassiou & Kotropoulos (2014) traitent cette question dans le cadre de la mise à jour du modèle qui sera repris ultérieurement.

En nous référant au modèle linéaire, nous avons tenté de vérifier si $p(z|q)$ pouvait être approché par la moyenne des $p(z|w_j)$ pour w_j dans *q*. Avec cette approche « théoriquement problématique » , les documents proches des documents test sont donnés dans le tableau 11.

Document	Sans *tf-idf*	Avec *tf-idf*
3d18 (corpus)	3d18 pr17 gp39 pr2 gp34 0.626 0.599 0.586 0.585 0.559	3d18 op88 fn11 op67 3d14 0.868 0.850 0.837 0.709 0.674
ud104	gp72 gp111 gp115 gp26 gp54 0.809 0.740 0.696 0.683 0.668	3d20 3d4 3d44 3d52 3d57 0.814 0.692 0.692 0.683 0.668
ud85	nu24 op93 nu15 nu21 nu4 0.915 0.912 0.904 0.886 0.868	gp6 gp127 gp48 gp26 gp21 0.881 0.866 0.805 0.790 0.788
ud87	op47 nu5 op90 lr20 op76 0.587 0.573 0.571 0.566 0.550	op83 pr34 nu12 gp42 al15 0.550 0.545 0.543 0.539 0.535
Req 1	gp131 gp16 gp103 gp129 gp44 0.861 0.847 0.825 0.780 0.772	gm6 gm11 gp8 gp2 gm4 0.872 0.864 0.840 0.840 0.836
Req 2	op21 op6 op85 lr1 op5 op78 0.666 0.665 0.641 0.603 0.601	op87 lr23 op21 lr12 pr21 0.730 0.581 0.566 0.561 0.536

Tableau 11. Les cinq documents les plus proches de chaque document test (selon *topics* pLSA, distance du cos)

documents. »

Commentaire

Pour le document 3d18, les composantes en mots est connu (c'est la dixième ligne de la matrice [documents x mots]. Pour les autres il s'agit tout d'abord de chercher les vecteurs correspondant en portant sur la liste des mots initiaux (le dictionnaire) les fréquences des mots correspondant du nouveau document.

Il n'y a pas de recoupement entre les requêtes avec ou sans transformation *tf-idf*. Les résultats en utilisant la distance de *Hellinger*, qui semblerait mieux adaptée, sont semblables. Il en va de même si l'on introduit dans le calcul une pondération qui tient compte des probabilités des *topics*. Cette divergence, vraisemblablement liée à notre procédure « théoriquement problématiques », va demander un examen plus approfondi.

ud104 (Les rubans transparents) : gp72 (Le carré de Joseph) présente une similitude de situation bien que les tâches sont différentes. La réponse 3d20 (L'octaèdre) est moins bonne.

ud85 (Le plus grand produit) : nu24 (Les tampons d'Emmanuelle et de Luc) présente une similitude de situation bien que les tâches sont différentes. La réponse gp6 (La saga des carrés) est moins bonne.

ud87 (Menteur et menteur) : la réponse op83 (Carrés magiques multiplicatifs) est moins bonne que op47 (Les châtaignes de Charles (II)) bien que cette dernière porte sur des raisonnements numériques alors que ud87 purement logique

req1 (rectangle, périmètre, aire, quadrillage) : les deux résultats gp131 (Comparaison de figures) et gm6 (La boucle (II)) ne sont pas très bons.

req2 (dénombrer, combinatoire, ordre, déduire) : les résultats obtenus avec cette requête sont assez déroutants. On s'attend à trouver plus de problèmes du domaine LR (Logique et raisonnement). Toutefois, la solution op6 n'est pas totalement incompatible. La réponse op21 (Vacances d'hiver) est meilleure que op87 (Menteur et menteur).

Pour mémoire : La perplexité calculée sur ces résultats à l'aide du package *text2vec* vaut : sans *tf-idf* 63.93 avec *tf-idf* 67.99

Conclusion intermédiaire

Un problème évoqué est le sur-apprentissage généré par cette méthode. Le modèle est adapté au corpus qui l'a généré mais ne serait pas utilisable pour de nouveaux documents. Blei & al (2003) évoquent notamment ce problème qui les a conduits à proposer le modèle LDA. C'est la croissance du nombre $KV+KM$ de paramètres à estimer, qui croît linéairement avec le nombre de documents, qui encline à ce sur-apprentissage. L'algorithme tempéré (TEM) utilisé pour lisser les paramètres du modèle pourrait permettre des performances prédictives acceptables. Mais il a été montré qu'un sur-apprentissage peut avoir lieu même dans ce cas (Popescul & al, 2001). Ces derniers auteurs proposent deux nouvelles méthodes qui améliorent la performance d'apprentissage. L'article en question reste à être étudié plus attentivement.

Il est vrai que le nombre de paramètres du modèle croit linéairement avec la taille du corpus d'apprentissage a tendance a figer le modèle. Toutefois, l'implication évoquée de sur-apprentissage ne semble pas rigoureusement établie. De façon générale, le lien entre dimension du corpus (nombre de documents et grandeur du dictionnaire) et nombre de *topics* n'est jamais abordé théoriquement. Cette discussion sera reprise ultérieurement.

Pour comparer LSA et pLSA, Hofmann (1999) procède à une probabilisation de LSA afin d'utiliser le coefficient de *perplexité* pour la comparaison. Il note qu'en général pLSA gagne en précision en utilisant une combinaison linéaire convexe d'un score brute de similarité et d'un score calculé dans

l'espace latent (voir *pertinence*).

Mise en oeuvre de pLSA en Python (pLSA Py)

Un programme en Python trouvé sur *Github* semblait implémenter les propositions de Hofmann. Mis en route et adapté en deux versions par Alain Favre, il semblait utile de l'ajouter dans la panoplie de nos tests sans entrer dans les détails et d'en faire la comparaison avec les autres méthodes en se cantonnant au cas sans transformation *tf-idf*. Le programme fournit les *topics* dans l'ordre des poids décroissants (sans les poids), par exemple pour le *topic* 0 : ['triple', 'recevoir', 'partager', 'moitié', 'double']. Le cas asymétrique étant traité, les documents sont donnés avec les *topics* les plus importants les concernant (sans valeur chiffrée). Par exemple : 3d1 : 39 31 25 40 11.

Les cinq documents proches du document 3d18 appartenant au corpus sont 3d22, gp38, gp2, 3d59, 3d19. Au vu du résultat concernant le document de contrôle cette procédure, répétée à deux reprises (pLSA Py2, pLSAPy'), selon les deux versions ne donne pas de bons résultats. Seule une famille de résultats sera utilisée pour mémoire dans la comparaison finale.

Mise en oeuvre de pLSA en Python (pLSA Z)

ZhikaiZhang propose également une implémentation de pLSA en Python utilisant l'algorithme EM (plusieurs minutes pour chaque itération, indice d'un nombre conséquent d'itérations). Par contre, aucune indication n'est donnée concernant la possibilité d'indexer un document hors du corpus.

L'algorithme délivre les matrices brutes (charge à l'utilisateur d'y ajouter des labels) : [documents x *topics*] et [*topics* x mots] de même que les mots principaux de chaque *topic*. La version est asymétrique et ne délivre pas la probabilité des *topics*. En principe on devrait pouvoir les calculer à partir de la relation $(p(d_i, w_j)) = (p(d_i|z_k))\, diag(z_k)\, (p(z_k|w_j))$.

Les tableau 12 et 13 donnent respectivement les mots définissant les *topics* et les *topics* indexant les documents (cas asymétrique).

Topic 1	classer	avoir	gagner	pouvoir	élèver
	0.10	0.07	0.07	0.06	0.06
Topic 2	heure	minute	aller	horloge	retour
	0.16	0.09	0.06	0.06	0.05

Tableau 12. Quelques *topics* avec p(w|z)

3d1	T25	T8	T42	T16	T11
	0.70	0.13	0.05	0.04	0.03
3d18	T28	T21	T22	T15	T36
	0.34	0.28	0.18	0.10	0.09
gp36	T17	T26	T14	T48	T22
	0.41	0.21	0.11	0.11	0.07

Tableau 13. Quelques documents avec p(z|d)

On peut s'étonner que 3d1 et 3d18 ne possèdent pas un *topic* important en commun. En consultant les fiches, on constate que le problème 3d1 est centré sur la visualisation 3d alors que 3d18 est centré sur un comptage où la visualisation joue un rôle mineur. Problème résolu !

Les distances avec les documents test sont calculées selon notre procédure approchée « théoriquement problématique ». Elles figureront dans la comparaison de l'ensemble des analyses.

4.3.3.2. LDA

Présentation

Par rapport à l'analyse précédente, l'analyse LDA, Latent Dirichlet allocation, ajoute pour les *topics* une distribution a priori de Dirichlet. Cette évolution est abondamment commentée dans l'article qui

semble avoir popularisé la méthode (Blei, Ng & Jordan, 2003)[1]. Une présentation qui propose un classement des méthodes apparentées se trouve dans la thèse de Ponweiser (2012).

Nous décrivons le processus génératif avec les paramètres proposés par Blei & al (2003) en améliorant la présentation formelle en nous inspirant de Gaussier & Yvon (2011).

M documents vont être générés à partir d'un vocabulaire de *V* mots. Sont également donnés :

- *K topics*.

- Un hyper-paramètre (ou méta-paramètre) α , vecteur de *K* nombres positifs

- Pour chaque *topic* z_k une distribution $(\beta_{kj})_j$ définissant $p(w_j|z_k)$ probabilité que le terme w_j soit associé au *topic* z_k ou le poids de w_j dans z_k .

  α et la matrice $\beta=(\beta_{ij})$ de dimension *K* x *V* seront estimés selon diverses méthodes[2].

Construction du document d_i pour $i=1,...M$:

- Eventuellement choisir la taille N_i du document en suivant par exemple une loi de Poisson

- Choisir une distribution de *topics* propre au document : $\theta_i \sim Dir(\alpha)$ avec $\sum \theta_{ik}=1$

- Pour *j* de 1 à N_i

  - Choisir un *topic* z_k suivant la loi Discrète(θ_i)

  - Choisir un terme w_j selon β_k

Selon les auteurs, le processus admet plusieurs variantes. Plus avant dans l'article de Blei & al (2003) ou dans la construction proposée par Steyvers & Griffiths (2007) ou encore Blei (2009) et Blei & Lafferty (2009) la matrice β est remplacée par un vecteur qui sert d'hyper-paramètre pour la distribution de Dirichlet Dir(β). C'est notamment le processus décrit par Gaussier & Yvon (2011). Cela diminue le nombre de paramètres sans que cette caractéristique ne soit jamais clairement évoquée.

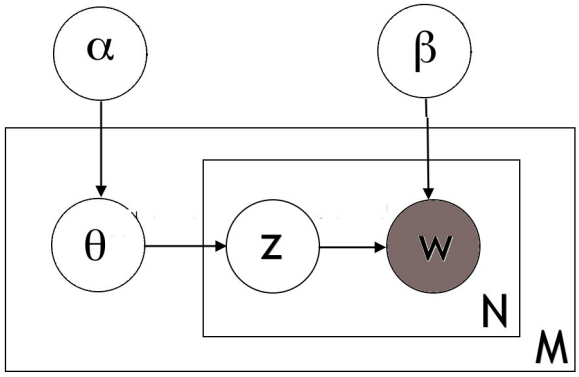

fig 2. Modèle graphique de LDA version Blei & al (2003)

L'estimation des paramètres peut se faire en maximisant le coefficient de vraisemblance et donc en minimisant le log likehood : $\sum_{i=1,..M} \log p(d_i|\alpha,\beta)$ selon l'algorithme EM mais aussi des techniques plus élaborées tels que l'échantillonnage de Gibbs[3].

1 Pour une présentation générale voir Blei (2012).

2 En partie des choix, en partie de l'apprentissage. Ce point sera précisé ultérieurement.

3 S'inspirant de Steyvers & Griffiths (2004), Brooks (2015) développe pas à pas une version élémentaire.

Une fois les $K + KV$ paramètres établis, les poids (probabilités) $p(z_k|d_i)$ peuvent être calculés pour chaque document d_i . Ce calcul peut-être prolongé à des documents hors du corpus.

Ce modèle connaît de nombreux perfectionnement (Murphy, 2012). En particulier Shu, Long & Meng (2009) évoquent les trois types de problèmes à résoudre lié au lien entre entité et terme pour la nommer : le problème du partage de nom (plusieurs entités peuvent avoir le même nom), le problème des variantes du nom (une entité peut avoir plusieurs noms) et le problème du mélange des noms plusieurs entités partagent plusieurs noms. Ils proposent des algorithme permettant de résoudre les trois types de problèmes. Ils se focalisent sur une extension de LDA (LDA-dual model) qui permet de traiter en entités séparées les mots et les auteurs.

Mazarura & Waal (de) (2016) explore un modèle moins populaire, le « Gibbs Sampling algorithme for the Dirichlet multinomial mixture model for short text clustering (GSDMM) », qui suppose que chaque document n'appartient qu'à un seul *topic*. Selon les auteurs, ce modèle est plus approprié pour les textes courts. Ce modèle est utilisé pour analyser trois corpus, l'un formé de longs textes et deux de courts textes (de type tweet). Les résultats fournis par LDA et GSDMM sont évalués selon l a *cohérence* et la *stabilité* (persistance des résultats en refaisant l'analyse). La *cohérence* est meilleure pour le long texte que pour un texte court. La *stabilité* est toujours meilleurs avec GSDMM[1].

Zhai & Jordan (2013) adaptent le modèle pour un vocabulaire infini.

Il reste toutefois un certain nombre de points à éclaircir pour distinguer les différentes approches. Cela concerne le nombre de paramètres, fixés, appris, les procédures d'apprentissage, l'intégration des documents hors corpus, le lien entre LDA et pLSA (cas particulier ou non?), etc. Toutes questions longuement débattues dans des forums. Sans compter que les packages adaptent aussi les aspects théoriques à leur manière. Nous y reviendrons dans la discussion.

Freitas & Barnard (2001), Chien, Wu & Wu (2005), Chien & Wu (2008) adopte une procédure quasi bayésienne pour pLSA en ajoutant une distribution a priori (Dirichlet) sur l'ensemble $\{p(w_j|z_k), p(z_k|d_i)\}$. Pour résumé, l'analyse pLSA standard s'en tient à maximiser le coefficient de vraisemblance (ML). En ajoutant une distribution a priori la procédure ajustée procède à une maximisation de la distribution a posteriori (MAP). Ce procédé rapproche cette version de pLSA (que nous noterons pLSA+) de LDA. La différence réside dans la manière d'introduire la distribution a priori. De plus, alors que LDA va jusqu'au bout du processus bayésien en utilisant la distribution prédictive a posteriori, pLSA+ passe par une estimation quasi bayésienne. Les implications des différences entre les deux procédures restent à examiner. Nous n'avons pas trouvé d'implémentation de cette variante de pLSA+.

Première mise en oeuvre de LDA (LDA tm)

Le premier package R utilisé est *topicmodel*s (Grün & Hornik, 2011, 2018)[2]. Il nécessite de transformer les matrices [documents x mots] en des objets de classe *DocumentTermMatrix*. L'analyse ne peut se faire que sur des fréquences absolues. La documentation fait référence à Blei & co (2003) et à un travail de Xuan-Hieu Phan et co-auteurs pour l'échantillonnage de Gibbs.

La procédure principale délivre la matrice des poids en *topics* de chaque document $gamma=(p(z_k|d_i))_{i,k}$ (dim 468 x 50) et la matrice $beta=(\log p(w_k|z_i))_{i,k}$ (dim 50 x 718) c'est-à-dire le logarithme de la distribution des mots pour chaque *topic*. On trouve également la valeur de l'hyper-paramètre α ajustée (0.1 par défaut).

1 Yin, J. & Wang, J. (>2013) proposent le modèle GSDMM pour de l'analyse cluster. La méthode peut inférer le nombre de clusters automatiquement avec un bon équilibre entre homogénéité et complétude.

2 Le package lda (Chang) d'abord envisagé s'avère ne pas traiter les multiplicités des mots.

Sans autre argument que le modèle, la procédure *posterior* crée les matrices $terms = (p(w_i|z_k))$ (50 x 718) c'est-à-dire le poids en mots pour chaque *topic* (exponentielle de la matrice *beta*) et la matrice *topics*, c'est-à-dire la matrice *gamma* mise en forme (de classe *matrix* avec nom des lignes et colonnes, etc.).

En ajoutant comme paramètre une liste de documents hors corpus, la procédure renvoie les poids en *topics* de chaque document. C'est la procédure initiale qui est utilisée avec le modèle comme paramètre supplémentaire. Le paramètre de contrôle est identique sans le calcul de la matrice *beta* récupérée du modèle. Pour en savoir plus il faudrait consulter le code C/C++ appelé.

Le package permet de calculer la perplexité qui vaut 81.22. Les données de l'analyse donnent une perplexité de 66.74 avec le package (décrit ci-après) *text2vec*.

Proximité de documents : via facteurs latents

Après avoir transformé le vecteur des documents en objets de la classe *DocumentTermMatrix* auquel on applique la procédure *posterior*. Les résultats obtenus pour les documents test sont donnés dans le tableau 14 où figure également une comparaison entre distance du cosinus, la divergence de Jensen–Shannon (KL symétrisée) et la distance de Hellinger. S'agissant de distributions à comparer, les distances JS et de Hellinger semblaient plus appropriées. Elles semblent peu considérées dans la littérature.

Document	Distance cos	Distance JS	Distance de Hellinger
3d18 (corpus)	3d18 pr16 3d4 3d57 3d26 1.00 0.97 0.92 0.88 0.85	3d18 pr16 3d39 3d4 3d57 0.00 1.29 1.69 1.81 2.18	3d18 fn6 pr16 fn2 gm8 0.00 0.11 0.26 0.29 0.49
ud104	gp125 gp115 gp94 gp28 gp72 0.82 0.79 0.79 0.79 0.77	gp125 gp115 gp28 gp68 gp94 2.49 2.97 3.22 3.24 3.38	
ud85	nu30 nu4 nu27 nu18 op39 0.83 0.80 0.80 0.79 0.79	nu30 nu23 fn9 lr15 nu4 2.36 2.45 2.56 2.75 3.14	
ud87	op75 op92 nu10 op101 lr24 0.80 0.84 0.84 0.83 0.83	pr21 lr1 op75 lr23 al8 2.14 2.27 2.40 2.42 2.52	
Req 1	gp2 gp8 gp3 gp99 gp75 0.90 0.90 0.90 0.88 0.87	gp44 gp2 gp8 gp58 gp3 1.11 1.74 1.74 2.18 2.26	gm15 gp94 fn2 fn6 gp74 0.50 0.51 0.55 0.56 0.58
Req 2	lr20 lr18 lr12 lr5 lr9 1.00 1.00 1.00 1.00 1.00	lr20 lr5 lr1 lr12 lr9 0.30 0.33 0.36 0.38 0.39	

Tableau 14. Les cinq documents les plus proches de chaque document test avec les distances du cos, JS et Hellinger

Commentaire

Les différentes distances s'accordent avec plus ou moins de bonheur selon les documents. Si nous avons adopté la distance du cosinus alors que JS paraît plus appropriée, c'est pour nous aligner sur plusieurs packages où la distance du cosinus est adoptée. Une brève évaluation des réponses est donnée en même temps que pour celles obtenues avec le package *text2vec* (tableau 15).

Utilisation du package 'text2vec' (LDA t2v)

Ce package (Selivanov, Bickel & Wang, 2020) qui est apparu après le début de notre quête a été le

dernier à être utilisé. Il inclut les outils de visualisation et d'évaluation que nous avions dû adapter nous-mêmes pour les autres systèmes. Il se réfère du point de vue théorique à Roeder, Both & Hinneburg (2015). Nous avons profité de cette nouvelle mise en route de la machine pour comparer les résultats obtenus avec 20 *topics* à ceux obtenus avec 50 *topics* (tableau 15).

Le modèle est élaboré avec l'algorithme WarpLDA (en C/C++) en établissant la distribution des mots sur les *topics* (correspond à la matrice de *beta* de Hofmann ou à l'exponentielle de la matrice *beta* du package *topicmodels*). La procédure *fit_transform* donne la distribution des *topics* sur les documents et t*ransform* les *topics* sur de nouveaux documents.

Document	50 topics	20 topics
3d18 (corpus)	3d18 fn19 pr20 lr19 gp37 0.81 0.64 0.61 0.61 0.57	3d18 op101 pr34 nu9 op95 0.90 0.82 0.82 0.82 0.80
ud104	gp115 gm5 3d40 gp26 gp21 0.70 0.66 0.66 0.62 0.61	gp28 gp107 gp9 gp56 gp119 0.88 0.86 0.83 0.82 0.80
ud85	nu15 op51 nu21 nu24 nu13 0.84 0.80 0.77 0.77 0.76	nu24 lr15 nu30 op7 nu14 0.92 0.92 0.92 0.88 0.86
ud87	pr8 pr23 lr18 op95 lr23 0.81 0.79 0.72 0.72 0.68	pr4 op50 op74 pr14 pr3 0.92 0.90 0.89 0.88 0.88
Req 1	gp8 gp3 gp15 gp131 gp29 0.82 0.82 0.78 0.75 0.74	gp48 gm5 gp85 gp47 gp21 0.97 0.95 0.95 0.95 0.94

Tableau 15. Les cinq documents les plus proches de chaque document test dans les modèles avec 50 et 20 *topics*

Commentaire

- ud104 (Les rubans transparents) : gp125 (Le plus beau parallélogramme) est une bonne réponse alors que gp115 (Les deux rectangles) n'est pas très satisfaisante de même que gp28 (Sur le mur de l'école (I))

- ud85 (Le plus grand produit) : Ce sont les plaques qui a rapproché ces réponses : nu30 (Tout à moins de 3 euros), nu15 (Drôle de nombre), nu24 (Les tampons d'Emmanuelle et de Luc). La dernière réponse est la meilleure.

- ud87 (Menteur et menteur) : op75 (Concours de pêche), pr8 (Perles rouges), pr4 (Le troc) sont apparentées pour des questions de logique, pr21 (Toujours le double) est moins bonne.

- req1 (rectangle, périmètre, aire, quadrillage) : gp2 (La table de jardin), gp8 (La boîte), gp48 (Les cinq carrés) sont apparentées, gp44 (Ornement grec) est moins bonne.

- req2 (dénombrer, combinatoire, ordre, déduire) : lr20 (Le pâtissier) pas trop mauvaise.

On s'aperçoit que la restitution est bien sensible au nombre de *topics* adopté dans le modèle, sans qu'il soit possible de déceler une solution meilleure que l'autre. De même, les deux algorithmes se valent du point de vue qualité.

Les coefficients de *perplexité* sont resp. 93.13 et 114.13 pour les modèles avec resp. 50 et 20 *topics*. Quant à la *cohérence*, les valeurs maximum et minimum sont données pour différents scores dans le tableau 16.

Modèle	logratio	pmi
50 topics, valeur min	-10.05	-7.34
50 topics, valeur max	-0.78	1.41
20 topics, valeur min	-8.22	-5.82
20 topics, valeur max	-1.09	1.20

Tableau 16. *Cohérence* minimum et maximum pour deux scores

Dans la prochaine partie quelques *topics* seront examinés de ce point de vue à l'aide de représentations graphiques.

LDA en Python (LDA Py)

Cette analyse suit en partie l'exposé de Li (2016) qui met en oeuvre les outils de la bibliothèque *Gensim*, sans le pré-traitement puisque déjà réalisé en Prolog et après avoir retransformé la matrice [documents x mots] en la liste des documents. Le coeur de l'estimation est basé sur un algorithme incrémental proposé par Hoffman, Blei & Bach (2010).

L'analyse s'est faite à deux reprises (notées Py et Py2) avec les paramètres par défaut (dont 50 itérations).

Le module Python *gensim* propose des mises en page élaborées. Par exemple le premier *topic* pour le document 3d2 :

```
Score: 0.35007521510124207
Topic:  0.092*"face"  +  0.049*"cube"  +  0.035*"figure"  +  0.034*"être"  +
0.032*"voir"
```

Proximité de documents : via facteurs latents

Document	Py sans *tf-idf*	Py avec *tf-idf*	Py2
3d18 (corpus)	3d18 3d25 3d64 3d7 3d6 1.00 0.74 0.68 0.68 0.68		
ud104	gp2 gp8 gp47 gp108 gm10 0.85 0.82 0.82 0.78 0.77	gp104 gp127 gp14 gp83 gp45 0.99 0.99 0.99 0.99 0.99	gp28 3d48 gp115 gp29 gp9 0.96 0.93 0.93 0.92 0.89
ud85	nu19 nu18 op34 nu24 nu3 0.91 0.90 0.89 0.89 0.89	op34 nu8 nu15 op93 op4 0.99 0.99 0.99 0.99 0.99	nu15 nu19 nu21 nu25 nu23 0.82 0.76 0.76 0.76 0.76
ud87	op95 op62 op31 fn1 al4 0.87 0.87 0.87 0.85 0.85	pr14 lr20 op79 op21 op15 0.99 0.99 0.98 0.98 0.97	lr11 op24 op95 pr7 op75 0.83 0.82 0.80 0.78 0.77
Req 1	gp94 gp131 gp16 gp83 gp81 0.80 0.77 0.77 0.77 0.76	gp2 gp8 gp111 gp88 gp64 0.98 0.97 0.97 0.96 0.95	
Req 2	lr21 nu29 lr18 lr20 al9 0.75 0.75 0.66 0.66 0.66	al9 al7 fn5 op67 lr23 0.95 0.84 0.81 0.81 0.80	

Tableau 17. Les cinq documents les plus proches de chaque document test (distance du cosinus)

Commentaire

- ud104 (Les rubans transparents) : des trois réponses : gp2 (La table de jardin), gp104 (Intersection) gp28 (Sur le mur de l'école (I)), la deuxième présente le plus de similitude.

- ud85 (Le plus grand produit) : nu19 (Le chiffre le plus utilisé), op34 (Objectif 2013), nu15 (Drôle de nombre) sont apparentées.

- ud87 (Menteur et menteur) : les réponses op95 Les dragons) et lr11 (Le petit Poucet et ses frères) sont meilleures que pr14 (Une photo d'Afrique).

- req1 (rectangle, périmètre, aire, quadrillage) : les réponses gp94 (La tarte de Mamie Lucie) et gp2 (La table de jardin) sont apparentées.

- req2 (dénombrer, combinatoire, ordre, déduire) : la réponse lr21 (Une coupe de glaces avec des amis) est assez bonne contrairement à al9 (Le bouquet).

Conclusion intermédiaire

A première vue les résultats semblent assez instables et très sensibles au pré-traitement. On remarquera toutefois que les distances varient très peu ce qui peut expliquer cette sensibilité.

La *cohérence* moyenne vaut 0.40 pour l'analyse menée sans transformation *tf-idf*, et 0.385 avec données transformées par *tf-idf*.

La marche à suivre contient peu d'information sur les procédures adoptées et les différents aspects théoriques (les résultats avec et sans transformation *tf-idf* sont présentés sans autres commentaires).

Le coeur du code de l'estimation[1] est basé sur l'algorithme incrémental (« on-line ») proposé par Hoffman, Blei & Bach (2010). Il sert donc aussi bien pour l'analyse du corpus initial que la calibration de nouveaux documents et la mise à jour du modèle[2].

4.3.4. Représentation graphique

Les graphiques que l'on peut obtenir avec la package LDAvis (sur R et Python) sont utiles pour examiner et tenter d'interpréter les *topics*. La représentation graphique donne les deux premiers axes d'une analyse en composante principale basée sur la distances de Jensen–Shannon[3].

Les articles sont classés selon leur *pertinence* (*relevance*) selon une formule n'intégrant semble-t-il pas le log (cette définition figure dans la partie l4.3.1 consacrée à l'évaluation page 12). Nous avons chaque fois adopté la valeur de $\lambda = 1$, ainsi les mots sont classés selon leur poids (leur probabilité de figurer dans la définition du *topic*).

$$relevance(w, z) = \lambda\, p(w|z) + (1-\lambda)\frac{p(w|z)}{p(w)}$$

Pour $\lambda = 1$, le coefficient est le poids de *w* dans *z* et pour $\lambda = 0$ ce même coefficient est rapporté à la fréquence de $p(w)$.

La *saillance* mentionnée en marge ne semble pas être utilisée dans la représentation. Nous commentons les graphiques sans tenir compte d'une petite incohérence relevée ultérieurement (voir Le problème du partage de nom page 37).

La figure 3 représente le *topic* 13 de cohérence minimum (analyse *text2vec*, 20 *topics*). Dans un

1 Le code est copié de onlineldavb.py (Hoffman)

2 Canini, Shi & Griffiths (2009) proposent également tel algorithme « en-ligne ».

3 Différentes distances mesures semblent possibles mais nous n'avons pas pu les mettre en oeuvre.

certain sens, sa cohérence n'est pas au niveau des contenus mais d'une précision sur des actions multiples. Le *topic* 15 (analyse *text2vec*, 50 *topics*, figure 4) est aussi à remarquer de ce point de vue. Ainsi nos analyses auraient plutôt tendance à regroupe les mots par catégories (verbes, noms, ...) plutôt que par actions (par exemple : dessiner carré).

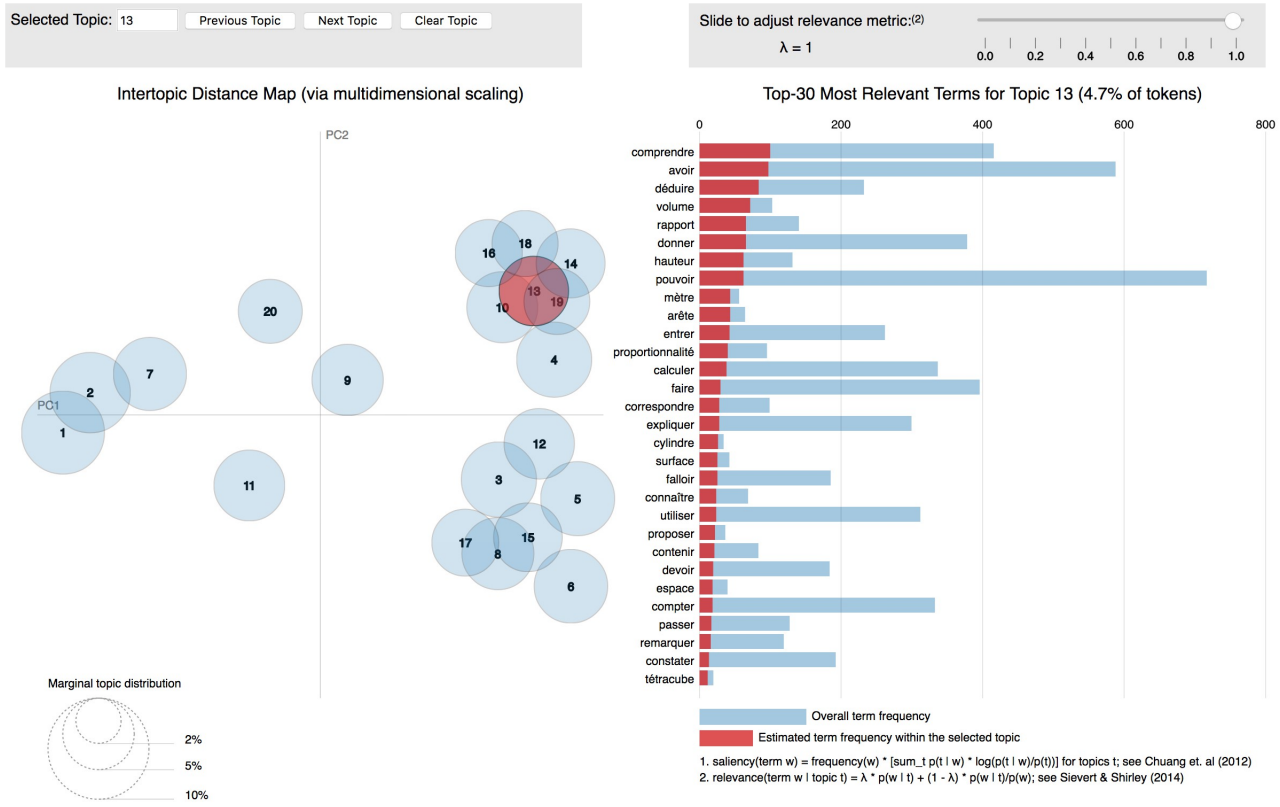

fig 3. *Topic* 13 (analyse t*ext2vec*, 20 *topics*)

Le *topic* 15 (figure 5) de cohérence maximum (analyse *text2vec*, 20 *topics*) semble être associé aux problèmes mettant en oeuvre les opérations arithmétiques. Le terme de poids maximum ne semble pas cohérent. Ce point sera à examiner en examinant les fiches dont le poids du *topic* 15 est maximum.

Les figures 6 et 7 illustrent des *topics* qui à première vue semblent bien cohérents. Elles correspondent à une deuxième analyse *text2vec* dans laquelle la répartition des *topics* est plus harmonieuse.

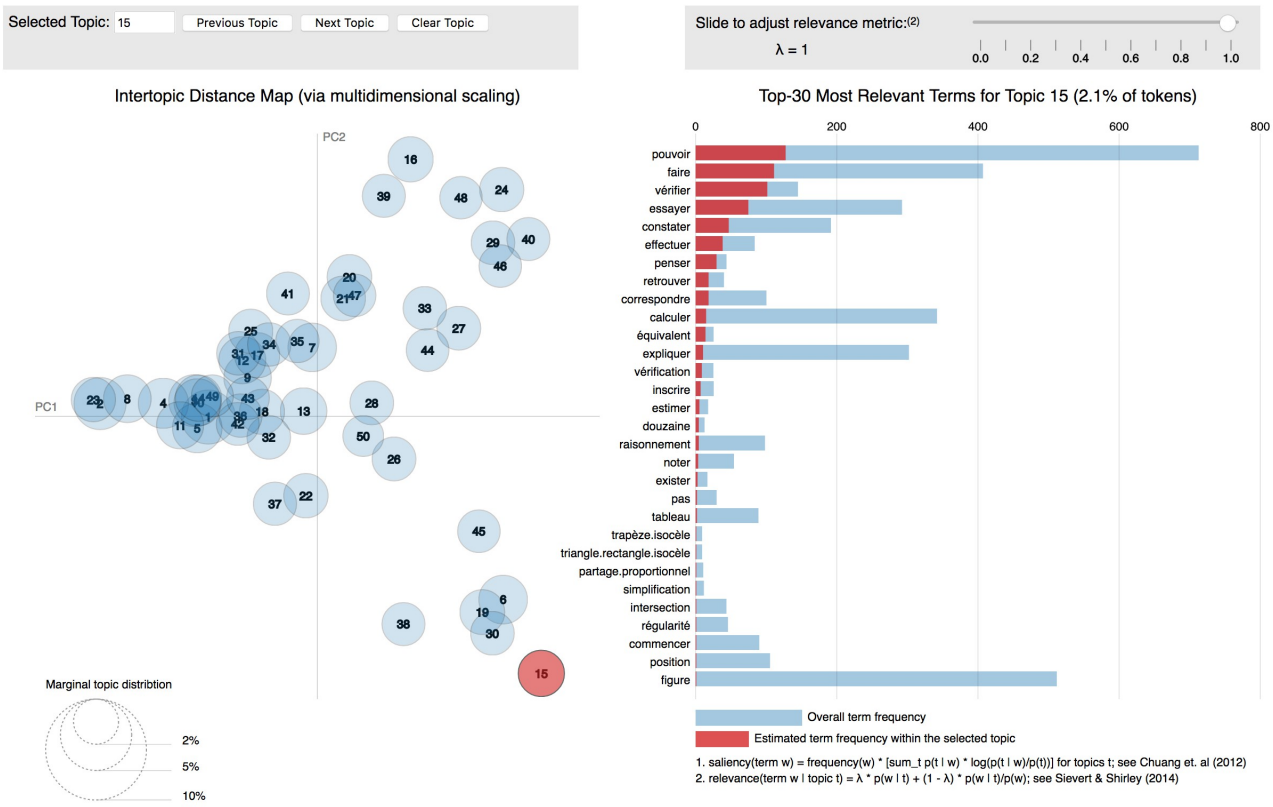

fig 4. *Topic* 15 (analyse *text2vec*, 50 *topics*)

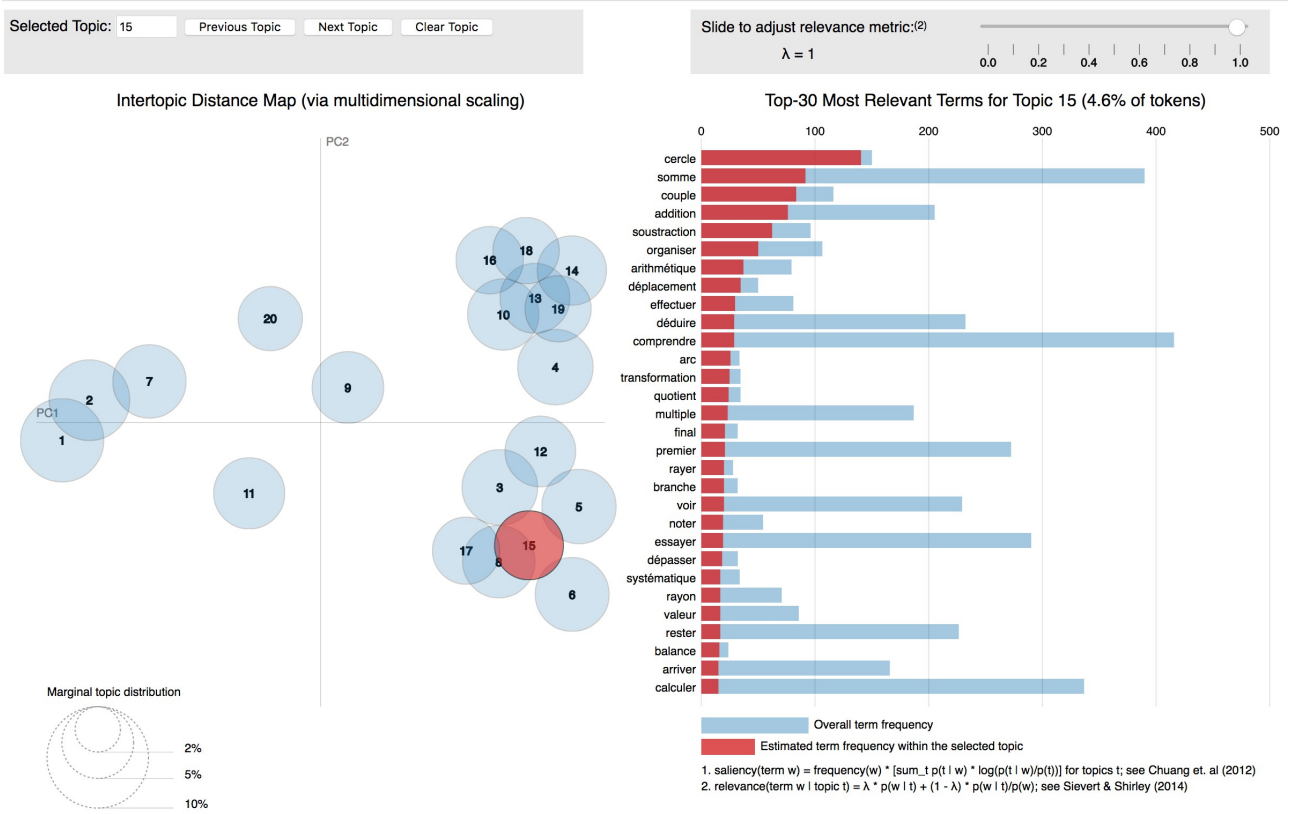

fig 5. *Topic* 15 de cohérence maximum (analyse *text2vec*, 20 *topics*)

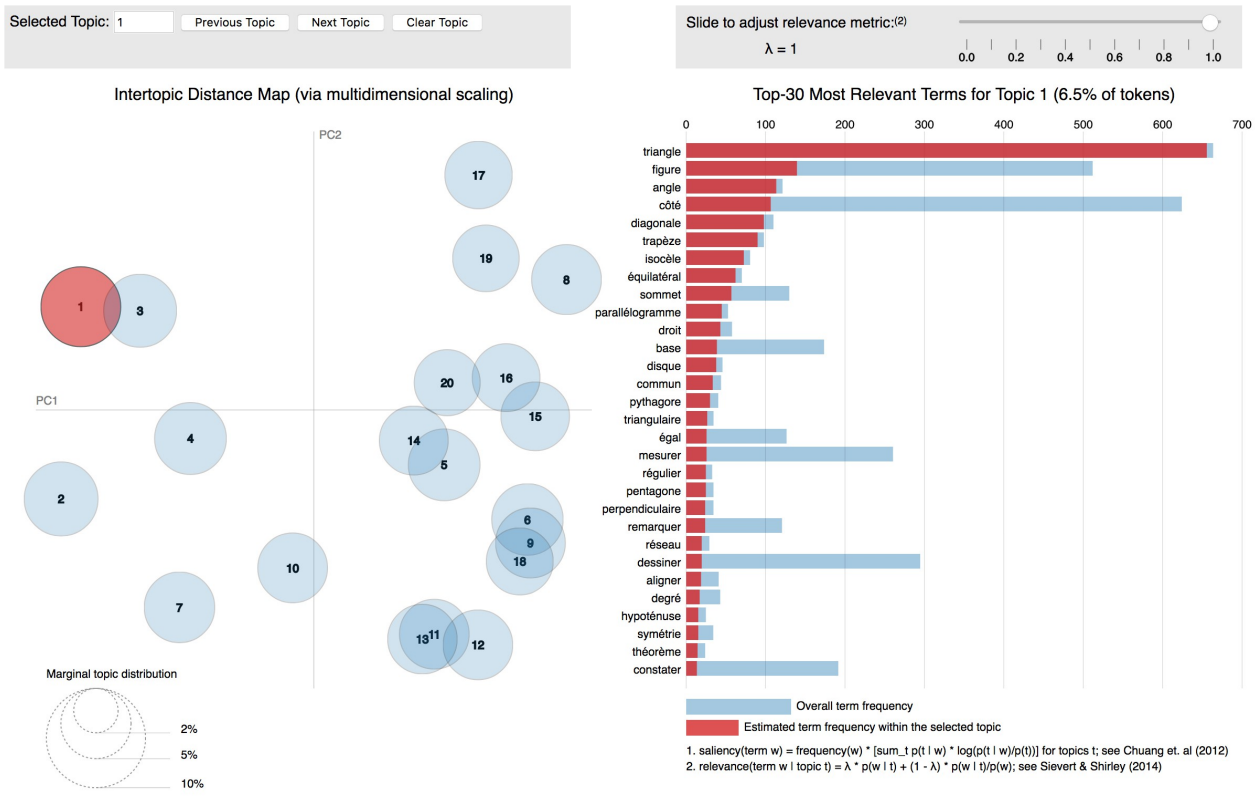

fig 6. *Topic* 1 (2e analyse *text2vec*, 20 *topics*)

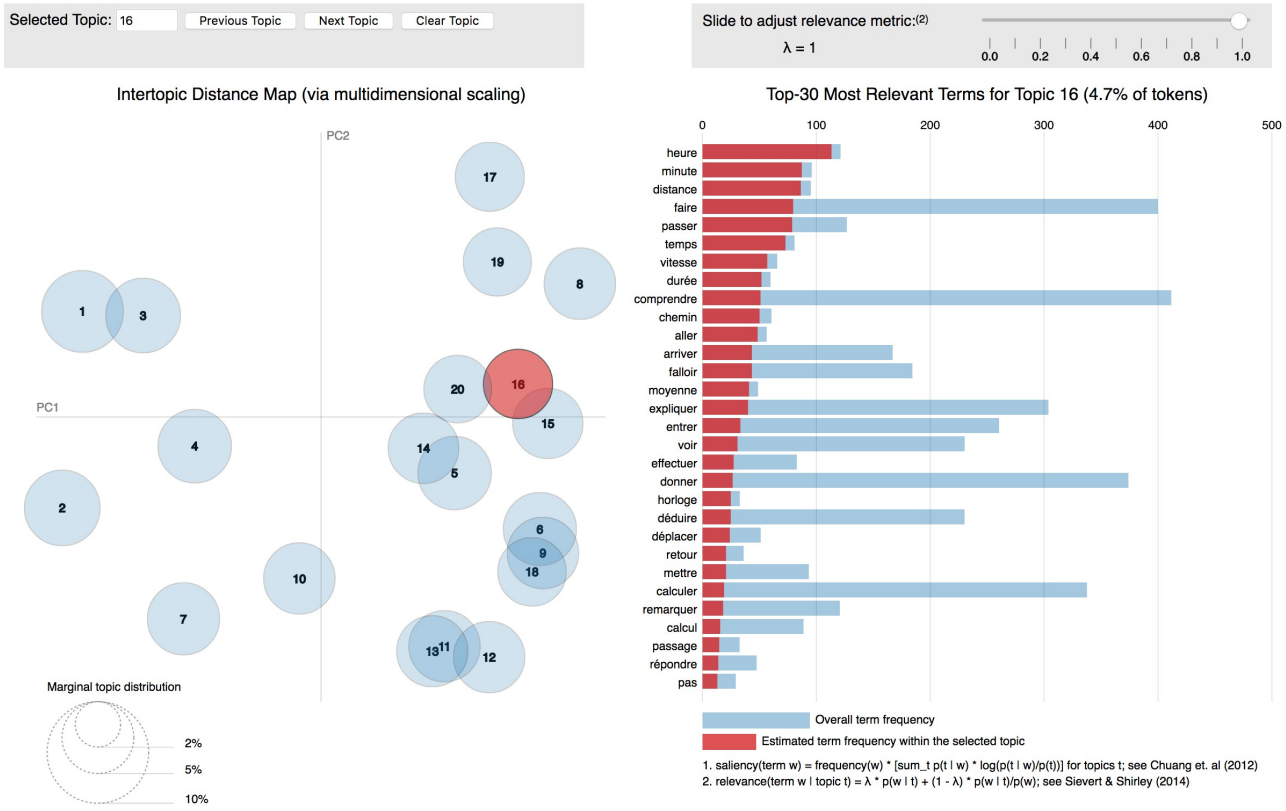

fig 7. *Topic* 16 (2e analyse *text2vec*, 20 *topics*)

4.3.5. Résumé des analyses

On verra par la suite que nos modèles, hormis le modèle déterministe LSA et les modèles pLSA Z et LDA t2v, ne sont vraisemblablement pas suffisamment « enseignés ». Ce qui limite la portée des comparaisons.

Document du corpus

Le tableau 18 propose une comparaison des documents proches de 3d18 (élément du corpus) déterminés directement (distance du cosinus) ou via la représentation en *topics*.

	Sans tf-idf	Avec tf-idf
Direct	3d18, lr19, gp126, lr21, 3d25 1.00 0.43 0.41 0.39 0.38	3d18, op27, pr20, gp34, gp37 1.00 0.21 0.21 0.20 0.16
LSA	3d18, lr19, al4, op80, op24 1.00 0.77 0.74 0.73 0.73	3d18, pr20, lr6, pr33, op2 1.00 0.76 0.76 0.67 0.63
LSA Q	3d18, lr19, op80, al4, nu24 1.00 0.70 0.64 0.56 0.54	
pLSA R	3d18, pr17, gp39, pr2, gp34 0.63 0.60 0.59 0.59 0.56	3d18, op88, fn11, op67, *3d14* 0.87 0.85 0.84 0.71 0.67
pLSA Py	3d22, gp38, gp2, 3d59, 3d19	
pLSA Z	lr13, 3d18, 3d19, pr11, op20 0.50 0.49 0.46 0.46 0.45	
LDA tm (cos/KL)	3d18, pr16, 3d4, *3d57*, 3d26 1.00 0.97 0.92 0.88 0.85 3d18, pr16, 3d39, 3d4, 3d57 0.00 1.29 1.69 1.81 2.18	
LDA t2v (20/50)	3d18, op101, pr34, nu9, op95 0.90 0.82 0.82 0.82 0.80 3d18, fn19, pr20, lr19, gp37 0.81 0.64 0.61 0.61 0.57	
LDA Py	3d18, 3d25, 3d64, 3d7, 3d6 1.0 0.74 0.68 0.68 0.68	3d18, fn10, op13, op38, op94 1.00 0.98 0.97 0.96 0.96

Tableau 18. Document proches de 3d18 (du corpus) selon les diverses analyses

Commentaire

Nous avons coloré les deux documents en tête de liste selon le code vert (bonne réponse), orange (acceptable), rouge (mauvaise)

On note que le document du corpus 3d18 (La boîte de sucres) arrive heureusement presque toujours en tête. Dans le cas de pLSA Z on constate un éloignement important, sans compter que le résultat

lr13 (Les bagues) n'est pas bon. Cela permet de remettre en cause notre procédure « moyenne » d'attribution des poids en *topics* aux documents hors corpus.

A noter que 3d19 (Les cubes) et 3d25 (Gourmandises) sont d'assez bonnes réponses alors que lr19 (Mathématiques dans la salle de gymnastique) pas du tout.

A noter que dans la banque des problèmes du type 3d18 (note :) sont finalement peu fréquents[1], sept seulement en ne tenant compte que de l'aspect 3d. Deux d'entre eux, marqués en italique figurent dans les réponses. Cela explique les distances assez grandes pour les documents qui viennent ensuite.

Documents hors corpus

Les tableaux 19, 20 et 21 proposent une comparaison des documents proches de respectivement de ud104, ud85 et ud87 (élément hors du corpus) déterminés directement (distance du cosinus) ou via la représentation en *topics*. Ils donnent le nombre de fiches communes pour les analyses prises deux à deux parmi les onze les plus proches.

ud104

	Brute s	Brute a	LSA s	LSA a	pLSA s R	pLSA a R	pLSA Py	pLSA Z	LDA tm s	LDA t2v	LDA Py s	LDA Py a	LDA Py2 s	Total
Brute s	X	9	9	2	4	0	0	3	3	7	2	2	3	44
Brute a	9	X	7	3	5	0	1	3	2	6	3	2	2	43
LSA s	9	7	X	2	4	0	0	4	2	7	3	1	2	41
LSA a	2	3	2	X	6	0	1	2	3	2	2	2	4	29
pLSA s	4	5	4	6	X	0	1	2	3	4	1	2	3	35
pLSA a	0	0	0	0	0	X	1	0	0	0	0	0	0	1
pLSA Py	0	1	0	1	1	1	X	0	1	0	3	1	0	9
pLSA Z	3	3	4	2	2	0	0	X	1	4	2	2	1	24
LDA tm	3	2	2	3	3	0	1	1	X	1	1	3	4	24
LDA t2v	7	6	7	2	4	0	0	4	1	X	1	1	1	34
LDA Py s	2	3	3	2	1	0	3	2	1	1	X	1	1	20
LDA Py a	2	2	1	2	2	0	1	2	3	1	1	X	2	19
LDA Py2 s	3	2	2	4	3	0	0	1	4	1	1	2	X	23
Total	44	43	41	29	35	1	9	24	24	34	20	19	23	

Tableau 19. Document proches de ud104 (hors corpus) selon les diverses analyses

1 Rappelons que les documents test ont été choisi au hasard.

ud85

	Brute s	Brute a	LSA s	LSA a	pLSA s	pLSA a	pLSA Py	pLSA Z	LDA tm	LDA t2v	LDA Py s	LDA Py a	LDA Py2 s	Total
Brute s	X	8	9	7	9	0	2	8	6	6	8	4	9	** Expression erronée **
Brute a	8	X	7	9	9	0	4	9	7	6	7	4	8	78
LSA s	9	7	X	6	8	0	2	8	5	6	7	2	9	69
LSA a	7	9	6	X	7	0	2	8	5	6	5	3	7	59
pLSA s	9	9	8	7	X	0	4	8	7	7	7	3	8	70
pLSA a	0	0	0	0	0	X	0	0	0	0	0	0	0	0
Plsa Py	2	4	2	2	4	0	X	2	3	4	3	1	3	42
Plsa Z	8	9	8	8	8	0	2	X	5	6	6	3	9	71
LDA tm	6	7	5	5	7	0	3	5	X	4	6	2	5	51
LDA t2v										X	4	1	6	
LDA Py s	8	7	7	5	7	0	3	6	6		X	4	7	59
LDA Py a	4	4	2	3	3	0	1	3	2		4	X	3	28
LDA Py2 s	9	8	9	7	8	0	3	9	5		7	3	X	62
Total	64	79	66	63	77	0	42	71	51		59	28	62	

Tableau 20. Document proches de ud85 (hors corpus) selon les diverses analyses

ud87

	Brute s	Brute a	LSA s	LSA a	pLSA s	pLSA a	pLSA Py	pLSA Z	LDA tm s	LDA t2v	LDA Py s	LDA Py a	LDA Py2 s	Total
Brute s	X	10	7	6	4	0	0	7	2	5	1	1	2	45
Brute a	10	X	7	6	4	0	1	7	2	6	1	2	3	51
Lsa s	7	7	X	7	4	0	0	7	2	4	1	1	1	41
Lsa a	6	6	7	X	4	0	0	6	2	2	2	0	1	36
Plsa s	4	4	4	4	X	0	1	2	2	2	1	1	2	29
Plsa a	0	0	0	0	0	X	1	0	1	0	0	1	0	3
Plsa Py	0	1	0	0	1	1	X	0	0	1	1	1	1	7
Plsa Z	7	7	7	6	2	0	0	X	1	2	1	2	0	35
LDA tm	2	2	2	2	2	1	0	1	X	1	1	0	1	15
LDA t2v	5	6	4	2	2	0	1	2	1	X	1	3	3	30
LDA Py s	1	1	1	2	1	0	1	1	1	1	X	0	4	14
LDA Py a	1	2	1	0	1	1	1	2	0	3	0	X	1	13

LDA Py2 s	2	3	1	1	2	0	1	0	1	3	4	1	X	21
Total	45	51	41	36	29	3	7	35	15	30	14	13	21	

Tableau 21. Document proches de ud87 (hors corpus) selon les diverses analyses

Commentaire

Par rapport aux constats précédents, en prenant les 11 documents les plus proches des documents d'évaluation, plutôt que cinq, la situation est un peu meilleure, la proportion du nombre de coïncidences augmente. Le degré de similitude entre les analyses varie selon les documents.

Les analyses les plus dépareillées concernent les analyses pLSA py et pLSA avec transformation *tf-idf*. Nous avons déjà exprimé nos doutes sur la pertinence de la première. Pour l'autre, nous prenons acte.

On notera une disparité des résultats selon les divers packages LDA. Cela nous ramène au problème du nombre de documents qui servent à constituer le modèle et au nombre d'itérations. Ce point est rarement signalé dans les articles. A noter que *text2vec* signale chaque fois le nombre d'itérations (160 pour 50 *topics* et 130 pour 20 *topics* sur les 1000 proposées par défaut) effectuées ce qui nous a permis de constater que nos premières analyses (qui prenaient souvent le nombre d'itérations proposées par défaut) étaient vraisemblablement trop faibles (50 pour LDA Py).

Les analyses dont les résultats comparatifs sont les plus proches de ce que l'on obtient avec les données brutes sont LSA, ce qui n'est pas une surprise et pLSA sans transformation *tf-idf* (avec notre bricolage), pLSA Z, LDA t2v. Les résultats sur ud85, pourraient conforter l'hypothèse du manque d'apprentissage pour certains modèles.

Requêtes 1

Le tableau 22 compare les concordances entres les diverses analyses pour la première requête.

	Brute s	Brute a	LSA s	LSA a	pLSA s	pLSA a	pLSA W s	pLSA W a	pLSA Py	pLSA Z	LDA tm s	LDA t2v	LDA Py s	LDA Py a	LDA Py2 s	Total
Brute s	X	8	7	6	6	2	4	3	0	7	3	8	4	2	3	63
Brute a	8	X	6	6	7	2	4	4	0	7	2	6	4	3	5	64
LSA s	7	6	X	6	6	3	4	3	0	6	4	6	2	1	2	56
LSA a	6	6	6	X	5	5	5	5	0	7	4	5	2	3	2	61
pLSA s	6	7	6	5	X	2	3	2	0	5	2	4	5	0	5	52
pLSA a	2	2	3	5	2	X	3	3	1	1	2	2	0	3	0	29
pLSAWs	4	4	4	5	3	3	X	1	1	3	3	5	2	1	3	42
pLSAWa	3	4	3	5	2	3	1	X	0	2	2	3	0	3	1	30
pLSA Py	0	0	0	0	0	1	1	0	X	0	1	0	0	0	0	3
pLSA Z	7	7	6	7	5	1	3	2	0	X	3	3	2	3	2	48
LDA tm	3	2	4	4	2	2	3	2	1	3	X	3	2	0	1	32
LDA t2v	8	6	6	5	4	2	5	3	0	3	3	X	3	1	4	53
LDA Py s	4	4	2	2	5	0	2	0	0	2	2	3	X	0	6	32
LDA Py a	2	3	1	3	0	3	1	3	0	3	0	1	0	X	0	20

LDAPy2 s	3	5	2	2	5	0	3	1	0	2	1	4	6	0	X	34
Total	63	64	56	61	52	29	42	30	3	48	32	53	32	20	34	

Tableau 22. Document proche de la première requête

Commentaire

La situation est similaire au cas précédent. A noter que la méthode pLSA W donne des résultats moins bons.

Le problème du partage de nom

Il y a plusieurs problèmes concernant le lien entre entité-notion et leur nom. Notamment plusieurs entités peuvent avoir le même nom. C'est le cas pour la notion de 'carré' qui peut signifier une figure géométrique ou l'opération d'élévation à la puissance 2. Lors de l'analyse *text2vec*, nous avons fait la liste des *topics* triée selon la valeur de $p(\text{carré}|z)$. Les deux *topics* dont ce poids est important sont les 18 (0.5) et 49 (0.31). En examinant ensuite ces deux *topics* on trouve que « carré » en est le terme de poids dominant comme on peut s'y attendre :

Topic 18 : carré (0.5) dessiner (0.06)

Topic 49 : carré (0.31) dimension (0.15) ... puissance (0.03)

Dans le premier cas, il s'agit du « carré » figure géométrique. Dans le deuxième, il s'agit de la notion de puissance qui apparaît toutefois assez timidement.

En voulant, représenter ce phénomène graphiquement à l'aide de *LDAvis*, nous constatons que les profils des *topics* 18 et 49 proposés dans le graphique ne correspondent pas à notre recherche directe.

Pour le moment ce mystère n'est pas éclairci. Il est fort possible que notre comparaison se base sur des « *runs* » distincts.

4.3.6. Mise à jour du modèle

La mise à jour du modèle est une opération importante pour les corpus qui évoluent, ce qui est le cas la plupart du temps.

On peut imaginer une refonte complète en utilisant comme paramètres d'initialisation des algorithmes les paramètres obtenus précédemment. Mais d'autres procédures plus économiques existent. Par exemple, lors de l'utilisation de l'algorithme EM, l'étape M peut utiliser une interpolation linéaire entre les *topics* existant et ceux dégagés par les nouveaux documents.

Chien, Wu & Wu (2005), Chien & Wu (2008) proposent une version quasi-bayésienne de pLSA qui permet d'utiliser un algorithme d'apprentissage incrémental. Cet algorithme permet d'adapter document après document aussi bien les paramètres que les hyper-paramètres.

Bassiou & Kotropoulos (2014) proposent également une méthode de mise à jour d'un modèle pLSA (online pLSA – oPLSA). A noter que les auteurs évoquent d'autres variantes du modèle pLSA (pLSA fold, par exemple) que nous n'avons pas examinées.

Hoffman, Blei & Bach (2010) proposent un algorithme variationnel bayésien pour LDA. Ils montrent que l'usage de cet algorithme sur un flux de document donne d'aussi bons résultats qu'une analyse sur l'ensemble du corpus en une seule fois[1]. Les procédures du package *Gensim* utilisent cet algorithme « incrémental », elles servent donc aussi bien pour l'analyse du corpus initial que la

1 Le module Python Gensim s'inspire utilise une partie de cet algorithme.

calibration de nouveaux documents et la mise à jour du modèle.

Les packages *text2vec* et *topicmodels* ne semblent pas proposer pas de procédure de mise à jour du modèle.

5. Discussion

Le problème à la base de ce parcours « initiatique » est de savoir s'il est possible de trouver un modèle permettant une classification de problèmes de mathématiques à la fois théoriquement satisfaisante, proche de la classification par tâche et utile pour la récupération (information retrieval).

Les deux objectifs principaux, outre le pré-traitement, de l'étude exploratoire qui font l'objet de cette chronique sont :

- Comparer différentes méthodes de réduction de la dimension de l'espace des problèmes et apprécier leur difficulté de mise en oeuvre.

- Evaluer quelle méthode semble donner les meilleures résultats par rapport au questionnement initial. En préalable, il s'agissait de dégager quelques critères utiles pour cette évaluation.

- Constater la possibilité d'une mise en relation de la classification « manuelle » par tâches et par *topics*.

Dans cette conclusion nous passons en revue les différents aspects du travail de la qualité des données à l'évaluation des résultats de quelques analyses. Sur cette base nous faisons le plan de notre futur programme.

Constat

Les données et le pré-traitement

Certains des problèmes rencontrés seront facilement résolubles. D'autres sont plus problématiques.

- Vocabulaire : Tout d'abord rappelons que nous avons limité le vocabulaire aux mots mathématiques (identifié grâce aux mot-clés) et aux verbes. L'usage a mis en évidence l'oubli de verbes et la nécessité d'enrichir le vocabulaire mathématique. Par ailleurs, certains vocables ont résisté au flitre (verbe avoir, par exemple).

- Problème des accents : il peut s'agir de problèmes orthographiques (*relève* à l'infinitif est *relever* et non *relèver*) ou de graphie (lorsque le verbe *établir* débute une phrase par *Etablir*, le passage en minuscule donne *etablir*).

- Transformation *tf-idf* : l'effet de cette opération n'est pas visible. Parfois elle peut améliorer la restitution, parfois non. Dans la comparaison sur les données brutes, LSA, LDA R les résultats sont équivalents. Les analyses pLSA (avec le bémol du calcul des profils des documents hors corpus) et LDA py seraient plutôt meilleures sans transformation t*f-idf*. Un pré-traitement de type 2 raisonnable serait encore à expérimenter.

- Mot-clés composés : certains mot-clés sont formés de plusieurs mots (p. ex. nombre naturel) qui sont rendus par un seul mot (p. ex. : nombre.naturel). Pour certains, le résultats de cette opération donne un mot peu fréquent alors que les composants seraient utiles pour caractériser le problème. Les solutions restent à explorer : statu-quo, introduction de chaque mot, etc. Le passage à l'utilisation systématique de bi-grammes seraient aussi une solution.

Mise en oeuvre de différentes méthodes

En général, l'installation et la mise en oeuvre des packages R ou des bibliothèques Python ne posent pas de problèmes majeurs.

C'est souvent la mise des données dans le bon format qui demande le plus de travail. En ce qui concerne les analyses elles-mêmes, les exemples fournis permettent de les mettre en oeuvre à un niveau basique. Par contre, la documentation est peu loquace pour mieux comprendre la fonction de divers paramètres. L'étude du code est souvent nécessaire pour déceler les variantes adoptées.

Aspect théorique

Les difficultés conceptuelles résident principalement dans la distinction des analyses pLSA et LDA.

Selon Blei & al (2003:1001) il est important de noter que dans $p(z|d)$ (pLSI), d est un index muet de la liste de l'ensemble d'entraînement. Ainsi d est une variable aléatoire multinomiale avec autant de valeurs possibles qu'il y a de documents et le modèle « apprend » le mélange *topics* $p(z|d)$ seulement pour les documents pour lesquels il est entraîné (sur-apprentissage). Pour cette raison, pLSI n'est pas un modèle génératif de documents bien défini ; il n'y a pas de manière naturelle de l'utiliser pour assigner des probabilités à des documents hors corpus.

LDA résout en principe ces deux problèmes connexes (sur-apprentissage et généralisation du modèle) en traitant les poids $p(z|d)$ par un paramètre K-dimensionnel donné par une variable aléatoire cachée plutôt que par un grand ensemble de paramètres liés à l'ensemble d'entraînement. Ainsi LDA est un modèle génératif bien défini qui se généralise facilement à de nouveaux documents. De plus, avec $K + KV$ paramètres (contrairement à $KM + KV$ pour pLSA) le modèle ne croît pas avec l'augmentation du nombre de documents. LDA ne souffre pas de sur-apprentissage comme pLSI.

La compréhension de cette différence fait couler pas mal d'encre sur les forums. En particulier sur Cross Validated[1] qui est un site de questions-réponses pour les personnes intéressées aux statistiques, au *machine learning*, *data analysis*, *data mining* et *data visualization*. Il bruisse de questions concernant LDA notamment de la différence avec pLSA.

L'auteur d'une question[2] commence par noter que les paramètres pour K *topics* pour l'analyse pLSA de M document et V mots sont au nombre de $KM + KV$ et que l'accroissement est linéaire en M ce qui a tendance à montrer que le modèle est sujet à sur-apprentissage.

Il précise en suite que LDA, selon la littérature : « ... supposé être un modèle génératif bien défini qui se généralise à de nouveaux documents. De plus le nombre de paramètres est supposé être $K + KV$ et donc n'augmente pas avec la dimension du corpus d'entraînement. »

Mais l'auteur de la note résume :

« Pour pLSA : $p(z|d)$ KM paramètres ; $p(w|z)$ KV paramètres »

« Pour LDA : $p(\theta_d|\alpha)$ KM paramètres (θ est de dimension K) ; $p(w|z)$ KV paramètres ; sans compter α et η les hyper-paramètres. »

« Ainsi le nombre de postérieurs à estimer est approximativement le même. Pourquoi alors est-il proclamé que LDA a résolu le problème de sur-apprentissage de pLSA ? Il est vrai que la distribution de Dirichlet avec α petit génère une distribution « rare », plus « rare » que $\alpha=1$ qui correspond à pLSA, et résout le problème de sur-apprentissage. Il n'en reste pas moins que le nombre de paramètres est le même » conclut-il !

1 https://stats.stackexchange.com/

2 https://stats.stackexchange.com/questions/155860/latent-dirichlet-allocation-vs-plsa

Le première réponse ne fait qu'un exposé général sans vraiment répondre aux questions posées.

« Le processus pLSA génère des documents avec une distribution de *topics* $p(z|d)$ dans un document particulier à l'opposé de générer des documents avec une proportion arbitraire de *topics* d'une distribution a priori. Cela n'est pas important pour la restitution d'une collection fixe. Mais dans des applications telles que la catégorisation de textes il est crucial d'avoir un modèle suffisamment flexible pour pouvoir intégrer proprement un nouveau texte ».

« Ainsi les probabilité des documents dans pLSI sont des points tandis qu'avec LDA ce sont tout un simplexe de *topics* (naturellement après entraînement on a une distribution de Dirichlet) ainsi il n'y a pas de problème pour intégrer de nouveaux documents. »

Le premier interlocuteur relance :

« Je ne comprends pas. Mais est-ce qu'ils n'ont pas le même nombre de paramètres ? De plus pLSA est identique à LDA avec le paramètre de Dirichlet $\alpha=1$ n'est-ce pas? Comment peuvent-ils être si totalement différents quand l'un est similaire à un cas particulier de l'autre ? »

A quoi le deuxième interlocuteur continue sur sa lancée : « LDA génère un hyperespace et est donc généralisable à des documents non vus. PLSA ne peut pas généraliser à des documents non vus et par conséquent ... ». Etc.

Le dialogue continue ainsi en restant parallèle[1]. Nous nous sommes aussi achoppés aux deux questions qui tracassent le premier interlocuteur : le nombre de paramètres et pourquoi un cas particulier d'un modèle ne possède pas les propriétés du cas général, notamment en ce qui concerne l'intégration de nouveaux documents.

Un autre interlocuteur vient à la rescousse, il précise : « ... par rapport à la question des paramètres devant être appris, dans LDA ce sont les *K* paramètres de Dirichlet [le vecteur α] duquel chaque θ_d sont déterminés. β est la matrice de KV paramètres construites tels que $\beta_{ij}=p(w_j|z_i)$. Autrement dit la *i*-ème ligne de β donne les paramètres de la distribution catégorique du mot *j* pour le *topic i* ».

« Ce sont les $K+KV$ paramètres référencés dans l'article et les seuls dont le modèle a besoin d'apprendre. Autrement dit chaque θ_d est tiré d'un Dirichlet avec le paramètre appris α ; il ne fait pas partie du modèle appris. Les packages LDA permettent de représenter un document comme une distribution sur les *topics* mais c'est une conséquence du modèle entraîné et non un prérequis. »

Cette réponse a le mérite de bien préciser quels sont les paramètres. Le premier interlocuteur reste cependant sur sa faim et conteste le fait que θ_d ne fait pas partie du modèle.

Sortons de ce dialogue pour résumer les trois problèmes :

- pLSA comme cas particulier de LDA ;

- nombre de paramètres ;

- calcul des poids en *topics* de nouveaux documents.

pLSA cas particulier de LDA

Si les démarches sont apparentées, pLSA n'est pas un cas particulier de LDA au sens où l'entend le premier interlocuteur ou que le note Nie (s.d.) lorsque le paramètre α est un multiple du vecteur 1.

A noter que le cas particulier $\alpha=1$ entre pas dans la définition générale de la distribution de

1 Un autre dialogue concernant ce sujet: https://stats.stackexchange.com/questions/242012/can-plsa-model-generate-topic-distribution-of-unseen-documents?rq=1

Dirichlet qui donne dans ce cas

$$p(\theta,\alpha)=\frac{\Gamma(K)}{\prod\Gamma(1)}\prod\theta_i^{\alpha_i-1}=(K-1)!$$

Implicitement, il est admis que cette distribution est la distribution uniforme (bizarrement, aucune indication à ce sujet n'est donnée dans les nombreuses présentations de la distribution de Dirichlet). Stricto sensu, le cas $\alpha=1$ n'est donc pas un cas particulier du cas général Dirichlet mais une distribution définie de façon ad hoc.

Ensuite pLSA se limite à la maximisation de la vraisemblance alors que LDA introduit une distribution a priori et procède à une maximisation a posteriori et introduit la distribution prédictive postérieur.

Cela n'implique pas forcément qu'il n'existe pas de méthode pour calculer, selon une méthode « théoriquement acceptable » le profil de nouveaux documents. Nous laissons cette question ouverte.

A noter la variante pLSA+ de pLSA basée sur une procédure quasi-bayésienne introduit également une distribution a priori.

Nombre de paramètres

La proposition du troisième interlocuteur du scénario rapporté ci-dessus « Les packages LDA permettent de représenter un document comme une distribution sur les *topics* mais c'est une conséquence du modèle entraîné et non un prérequis » ce qu'illustre d' ailleurs la formule de Blei & al (2003) pour la distribution prédictive a posteriori :

$$p(\boldsymbol{w}|\alpha,\beta)=\int p(\theta|\alpha)\left(\prod_{n=1}^{N}\sum_{i=1}^{K}p(z_i|\theta)\,p(w_n|z_i,\beta)\right)d\theta$$

Toutefois, en considérant les variantes possibles et les implémentations des algorithmes, la situation est plus embrouillées et la discussion concernant le nombre de paramètres passent au second plan.

En particulier le nombre de paramètres dans le cas où une distribution a priori est introduite sur le vocabulaire n'est jamais évoqué. Par ailleurs, selon le cas, les hyper-paramètres peuvent être fixés a priori ou estimé.

Pour pLSA+, les auteurs notent : « l'algorithme incrémental procède à l'estimation des paramètres comme des hyper-paramètres (Chien, Wu & Wu, 2005 ; Chien & Wu, 2008)

Gaussier & Yvon (2011) précisent que les valeurs de α,β,K permettent de moduler le comportement du modèle LDA . Il semble préférable d'estimer les valeur de α et β de plutôt que de les fixer à priori.

Griffiths et Steyvers traitent les hyper-paramètres α et β comme des constantes et les conséquences de certains choix sur le nombre de *topics* par document et le nombre de mots par *topic*. Des valeurs idéales sont proposées pour un bon compromis entre le nombre de *topics* par document et le nombre de termes par *topics*. Wallach, Mimno & McCallum (s.d.) précisent les choix à faire (plutôt asymétriques pour les documents et symétriques pour le vocabulaire). Syed & Spruit (s.d.) étudient également les conséquences de ces choix.

Dans le code Python le paramètre *alpha* peut-être fixé ou déterminé automatiquement (par analyse ou heuristique?).

Agrawal, Fu & Menzies (2018) dans une critique du modèle primitif de LDA propose une procédure complétée qui permet de déterminer K (nombre de topics), α et β (resp. hyper-paramètres pour la distribution de *topics* sur les documents et hyper-paramètres pour la distribution

des termes sur les *topics*) de façon à assurer une bonne stabilité à l'analyse.

Nous n'avons pas trouvé d'informations sur les effets de l'ampleur relative du vocabulaire, du nombre de documents dans le corpus et le nombre de *topics* et les heuristiques qui permettent d'éviter le plus possible les maximums locaux.

Calcul des poids en *topics* de nouveaux documents.

Le cas des nouveaux documents est plus problématique. Il est clair que LDA suivant le modèle bayésien complet, propose une distribution prédictive a posteriori, ce que pLSA ne fait pas. Nous ne sommes pas en mesure d'apprécier la proposition de Hofmann qu'il faudrait implémenter et tester.

Finalement, nous notons le choix de la distribution de Dirichlet est lié à la facilité (relative) des calculs offerte (Dirichet est la distribution a priori conjuguée à la distribution multinomiale), et non à des considérations linguistiques. Quelles seraient d'autres alternatives ?

Résultats

Stabilité

Selon les analyses les résultats sont assez disparates. Certaines analyses sont à exclure des outils à utiliser. Pour d'autres, il s'agira de modifier les paramètres proposés par défaut.

Un examen rapide des résultats donne lieu à deux impressions contradictoires. D'une part on constate une certaine convergence malgré des données relativement disparates. Une impression d'aléatoire subsiste, notamment entre les divers package LDA. Cela nous ramène au problème du nombre de documents qui servent à constituer le modèle et au nombre d'itérations. Ce point est rarement signalé dans les articles. A noter que *text2vec* signale chaque fois le nombre d'itérations (160 pour 50 *topics* et 130 pour 20 *topics* sur les 1000 proposées en paramètres) effectuées ce qui nous a permis de constater que nos premières analyses (qui prenaient souvent le nombre d'itérations proposées dans les démos) étaient vraisemblablement trop faibles. A voir si l'augmentation du nombre d'itérations permet améliorer une fidélité qui laisse à désirer pour l'analyse LDA Py.

Cette question ne se pose en principe pas pour les analyses déterministes.

Bien que nous ne pouvons pas comparer notre modèle réduit d'analyse aux travaux des experts du domaine, nous notons que Agrawal, Fu, & Menzies (2018) précisent que l'analyse LDA classique est assez instable et proposent un ajout correcteur.

Restitution

LSA montre une restitution qui semble la meilleure de toutes les analyses. Stevens & al (2012) notent que leur expérimentation montre que si LSA est le pire modèle en termes de score de *cohérence* de *topics*, mais qu'il produit une approximation stable, fidèle, précise comparable au jugement humain en terme de similarité.

On note que pLSA Z avec notre approche vectorielle et LDA t2v donnent, de ce point de vue des résultats semblables..

Il est intéressant de noter que la recherche par mot-clés en utilisant la req 1, ne conduit à aucun résultat alors que dans tous les autres cas les réponses existent et sont assez satisfaisantes. A noter que ce n'est pas lié aux modèles ; c'est aussi valable avec données brutes.

La proposition d'utiliser une moyenne pondérée entre la distance calculée sur les documents brutes et celle obtenue dans l'espace des *topics* et la distance du cosinus sur fréquence absolue n'a pas été explorée dans la mesure où pratiquement elle augmente la charge en calcul.

En définitive, s'agissant de la question de la restitution, l'intérêt de la réduction de l'espace par ces analyses semble discutable. La charge en calcul d'une comparaison directe ne semble pas différer beaucoup de celle nécessaire pour préalablement passer de la requête brute à sa version dans l'espace latent. Ce point serait à examiner plus précisément. Par contre, et particulièrement dans notre cas, la question de la classification est conceptuellement plus importante.

Classification et interprétation des *topics*

Malheureusement, il est encore trop tôt pour apprécier ce travail de ce point de vue. Dans certaines analyses un survol rapide montre des signes encourageants. C'est plus disparate dans d'autres.

Il semble certain que nous avions trop de *topics* et que la situation est trop touffue. Si les modèles déterministes, sont localement cohérents selon les auteurs cités, il n'en va pas forcément de même globalement. L'interprétation est dans tous les cas difficile certains *topics* semblent s'intéresser au contenu (triangle, aire, ...) d'autres aux actions (déterminer, calculer, ...) d'autres des mélanges hétéroclites.

Par ailleurs, des essais de mettre en relation classification par tâches et *topics* n'ont pas abouti à des résultats convaincants.

Pour cela, on a créé plusieurs matrices [documents x tâches] bordées d'une colonne supplémentaire fournie par une des classifications (LSA, pLSA, LDA, …) appliquées à la matrice [mots x documents]. Les étiquettes des classes sont les numéros des *topics* de poids maximum (chaque *topic* est une classe).

On a ensuite fait tourner des algorithmes classiques de discrimination supervisée. Les fonctions discriminantes cherchées devant reproduire la classification donnée par les *topics* : 80% des documents servent à l'apprentissage, 20% au test.

Divers modèles ont été utilisés en utilisant le package Python *sklearn* et les résultats sont de l'ordre suivant : Le 20 % seulement des documents de test sont discriminés correctement après l'apprentissage. Ces résultats sont comparables pour toutes les méthodes discriminantes utilisées pour obtenir les *topics* obtenus par LSA, pLSA, LDA.

On en conclut qu'on ne retrouve pas les *topics* par des analyses discriminantes standard et qu'il n'y a pas de lien clair entre les tâches qui caractérisent les documents et les *topics*. Cette conclusion laisse toutefois en suspens le fait que les *topics* peuvent conduire à une autre classification, possiblement meilleure, que la classification a priori par tâche qui comme déjà signalé pose problème.

En tout état de cause, il est trop tôt pour trancher davantage. L'ensemble des problèmes saturant l'un ou l'autre *topic* devait être examiné par des experts du domaine mathématiques scolaires.

Selon coefficients de perplexité et cohérence

Nous avons observé que ces coefficients, répondant à des définitions diverses dont nous n'avons pas cherché à vérifier l'équivalence, sont surtout utilisés pour déterminer le nombre adéquat de *topics*. Pour le moment nous nous sommes contentés d'engranger quelques valeurs « pour voir » et peut-être en vue de comparaison à des travaux ultérieurs. Nous serions surtout intéressés à comparer la qualité indiquée par ces indices à des jugements humains.

Cette remarque nous conduit à constater que la littérature fourmille de propositions de procédures d'évaluations automatisées grâce à des multiples indices qui sont jugés à l'aune de corpus standard, mais que l'on trouve rarement dans des travaux mettant en oeuvres ces techniques dans des projets pratiques.

A noter un usage inverse est proposé par Misra, Cappe & Yvon (2015) qui utilisent LDA pour détecter des documents sémantiquement incohérents.

Travaux futurs

Les données et le pré-traitement

Après avoir complété les fiches pour près de 1500 problèmes, les données seront à traiter après avoir compléter les verbes et augmenter le vocabulaire mathématique. L'analyseur est à améliorer si l'on veut garder les caractères accentués et non les supprimer comme on le pratique dans certains projets. Nous avons déjà signalé qu'un pré-traitement de type 2 raisonnable serait encore à expérimenter et que le traitement des mot-clés composés reste à décider.

Outre des analyses sur l'ensemble des fiches (résumé, tâche, consigne, concept), des analyses pourraient être faites, à fin de comparaison, séparément pour les rubriques résumé et tâche et d'y adjoindre une analyse sur les énoncés aux sens large (énoncé et consigne).

Mise en oeuvre de différentes méthodes

Nous continuerons à utiliser et comparer plusieurs algorithmes en maitrisant mieux les paramètres (nombre de *topics*, paramètre des distributions a priori, etc.). Nous gardons dans notre panoplie LSA (notre version), pLSA Z (avec une réflexion supplémentaire sur l'usage des documents hors corpus ou l'une ou l'autre des solutions « théoriquement peu satisfaisante »), LDA Py et LDA t2v.

Une attention sera portée sur la cohérence et la possibilité d'interprétation de chaque *topics*, ceci dans le cadre de la classification des problèmes. Mais nous avons bien compris qu'il nous fallait limiter le nombre de *topics*. Par contre, dans la comparaison avec d'autres index, il s'agirait de jouer avec la conjonction de *topics*.

Dans le registre pratique (restitution), l'indexage par topics (équivalent ou pas à d'autres indexations) devra être mis en oeuvre dans la banque de problème. Relativement simple en soi l'opération pourrait s'achopper à des problèmes de temps réel (analyse des requêtes et calcul de proximité).

A l'usage, il pourra être procédé à un affinement du modèle que ce soit sur le corpus de base ou lors de l'augmentation du corpus.

Un examen des modèles en partie supervisés sera également à mener dans le sens où plusieurs articles (Blei et al., 2004 ; Griffiths et al., 2004 ; Griffiths, Steyvers, Blei & Tenenbaum, 2005) évoquent des extensions des modèles permettant de mieux approcher le niveau sémantique. L'analyse *labeled LDA* est également une variante à envisager. Une synthèse entre procédures statistiques et ontologiques sera à trouver.

6. Pour conclure

Au début, il y avait l'analyse factorielle, celle qui pour coller aux « vecteurs de l'esprit », avait besoin des rotations. Débarrassée de son interprétation première, elle devint avec ses composantes principales un outil universel de réduction des dimensions. D'autres variantes suivirent.

Dans les années quatre-vingt-dix, un accueil chaleureux mais assez naïf est réservé à LSI. Nous avions notamment relevé dans le cadre de la recherche de documents l'article de Berry, Dumais & Shippy (1995) l'usage de LSI dans une application X-Windows pour la recherche de document (système LSIRS – Latent Semantic Indexing Retrieval System)[1].

Dessus & Lemaire (1999) sont encore plus optimistes et dans l'ambiance de la question « Les

1 Actuellement la question est aussi posée à propos de l'usage fait par LDA par le moteur de recherche de Google (https://www.scriptol.fr/seo/lda.php).

ordinateurs mettront-ils un jour les enseignants à la porte ? » présentent un système basé sur LSA qui devrait permettre de noter des dissertations par le biais d'un appariement sémantique de ces dernières avec des copies modèles sélectionnées au préalable par l'enseignant.

Depuis quelques années, celles où nous avions travaillé sur des documents caractérisés par des descripteurs et des références (Pochon & Favre, 2007)[1] dans une perspective plus ontologique, le nombre de travaux a explosé. Notamment en ce qui concerne l'indexage des documents et l'extraction de communautés dans des (grands) graphes.

Il fallait que nous nous mettions à la page. Le problème de la banque de problèmes de mathématiques nous en a fourni l'occasion. Mais nous ne savions pas ce qui nous attendait !

Nous sommes tombés dans le monde où la perspective déterministe des analyses s'ouvre à une vision statistique, les solutions analytiques font place aux solutions approchées obtenues par itération ou par échantillonnage sophistiqué, la classification de données à taille humaine s'élargit à des corpus gigantesques[2], la projection de nouvelles données est remplacée par des processus d'ajustement en continu, la théorie de classification de pattern s'applique à l'apprentissage des machines.

En quelques années des modèles statistiques se sont développés ce que la puissances de calcul et la quantité de données disponibles a rendu possibles et utiles. Ils ont chamboulé le paysage dessiné par les statisticiens traditionnels. Comme le note Cardon (2015), l'arrivée des data-analystes armés de leurs techniques de traitement massif des données a mis le monde des statisticiens en ébullition. On assiste aussi à une victoire posthume des statistiques bayésiennes sur le modèle fréquentiste jugé souvent plus objectif.

On peut parler d'une véritable fièvre. A côté de ceux de ténors, les travaux vraisemblablement d'étudiants prolifèrent qui proposent des marches à suivre utiles mais souvent limitées[3]. Il n'est pas impossible que différentes chapelles se disputent pour imposer leur meilleure méthode.

Toutefois, les concepts semblent encore peu stabilisés même si plusieurs algorithmes enrobent le même noyau. Le domaine est si jeune que bien des notions n'ont pas encore reçu de nom de baptême en français. Le temps est à l'expérimentation et à l'appropriation des ces techniques (des boîtes noires pour beaucoup d'utilisateurs) comme le furent les analyses factorielles ou la spectrographie. L'époque est à la botanique de données.

A noter qu'il est difficile de connaître exactement les implémentations des algorithmes généraux présentés dans les articles. Il faudrait se plonger dans le code C/C++ des quelques routines de bases auxquels les différents packages R ou Python font finalement appel.

Par ailleurs, la littérature classique propose des techniques avec des jeux de données standard et l'on rencontre rarement un usage en vraie grandeur où l'utilité mesurée dépasse le calcul de coefficients abstraits. Il y a peu de références aux contenus et aux usages. Ou alors les usages sont étudiés sans référence aux aspects techniques.

Il y a aussi des enjeux économiques qui limitent peut-être la diffusion des solutions adoptées chez les entreprises de l'information.

L'époque marque aussi un tournant en IA (IA faible) où les machines cherchent moins à modéliser le raisonnement qu'à ingurgiter des contextes à travers d'énormes masses de données. Plutôt qu'intelligentes elles deviennent statistiques.

1 Voir aussi http://www.projet-ermitage.org/thema/

2 Pas forcément textuels. Pour des modèles graphiques voir Seidenari (s.d.) ; Fei-Fei & Perona (2005) ; Sivic, Russell, Efros Zisserman & Freeman (2005). Les deux dernirs travaux sont résumés dans Malisiewicz (2006).

3 Tang (2019), Jones (2019), Doll (2018), etc. Certains sites en font une promotion: https://medium.com/ ; https://towardsdatascience.com , etc.

Porté à l'extrême cette recherche de régularité dans les données en faisant le moins d'hypothèses possibles est peut-être un puissant levier pour avancer dans la recherche scientifique. Mais est-ce vraiment la fin du modèle scientifique basé sur des théories à tester comme l'annoncent à grand fracas certains gourous de la Silicon Valley (en particulier Anderson, 2008) ? Les modèles seraient-ils devenus une sortie plutôt qu'une entrée. La recherche de corrélations sans se préoccuper d'élaborer un modèle a-t-elle un sens ?

On peut en douter ou, avec Postman (1993), le redouter. Les dangers sont grands que la science soit soumise à des algorithmes dont on sait qu'ils ne sont pas neutre dans leur apprentissage. Toutefois, ce thème ne peut se contenter de généralités et nécessiterait une discussion détaillée cas par cas.

Pour revenir à nos moutons précisons que nous oeuvrons encore dans le cadre des « small data ». Par rapport au but que nous recherchons, les résultats obtenus ne sont encore trop convaincants. Nous devons être plus sélectifs dans le vocabulaire et ajouter une certaine sémantique sous forme de bi-grammes. Une procédure également évoquée dans la littérature est d'introduire dans le corpus des documents calibrés pouvant faire le lien entre différents concepts (pour lier par exemple : carré et puissance ou carré et triangle).

De façon générale, il faudra nous assurer que notre notion de tâche est consistante et constitue une théorie sous-jacente (comme les vecteurs de l'esprit) qui justifierait un modèle non supervisé. Sinon cela pourrait nous inciter à nous tourner vers un modèle partiellement supervisé.

Il s'agit aussi de consulter les travaux proposant des modèles mixtes, intégrant modèles statistiques de données et ontologies[1]. Et, peut-être, en nous inspirant des travaux d'un séminaire Grize-Cardinet organisé à l'IRDP dans les années quatre-vingt, développer un modèle génératif de problèmes de mathématiques scolaires[2].

Cette chronique apporte davantage de questions que de réponses. Nous gardons toutefois l'espoir que nos tâtonnements puissent aider d'autres « amateurs » à entrer dans ce domaine en ébullition.

(Version du 30 septembre 2021)

Bibliographie

Ailem, M., Role, F. & Nadif, M. (2015). *Co-clustering Document-term Matrices by Direct Maximization of Graph Modularity*. Paris : LIPADE, Université Paris Descartes (CIKM_ailem.pdf)

Agrawal, A., Fu, W. & Menzies, T. (2018). *What is Wrong with Topic Modeling*? (*and How to Fix it Using Search-based Software Engineering*). arXiv:1608.08176v4 (WhatsWrongTopix.pdf)

Anderson, C. (2008). The end of Theory : the Data Deluge Makes the Scientific Method Obsolete. *Wired Magazine*, juin 2008.

Anonyme (s.d.). *L'échantillonneur de Gibbs* (cours3.pdf)

Bassiou, N. K., Kotropoulos, C. L. (à paraître). Online PLSA: Batch Updating Techniques Including Out-of-Vocabulary Words. I*EEE Transactions on neural Networks and Learning Systems*. (D48.pdf)

Berry, M.W., Dumais, S.T. & Shippy, A.T. (1995). *A case study of latent semantic indexing*. Knoxville: University of Tennessee, Computer science department. (10.1.1.48.1929.pdf)

Berney, J. & Pochon, L.-O. (2000). *L'Internet à l'école : analyse du discours à travers la presse*. Neuchâtel : IRDP (monographie 00.5)

1 Nous pensons à l'exemples des systèmes experts bayésiens où la connaissance a priori s'ajuste par apprentissage.

2 Les modèles de Shank avaient notamment retenus l'attention.

Bhatia, P.S & Lovleff, S. (2019). *A tutorial for blockcluster R package* (Version 4) (blockcluster_tutorial.pdf) https://rdrr.io/cran/blockcluster/man/cocluster.html

Bishop, C.M. (2006). *Pattern Recognition and Machine Learning*. Heidelberg : Springer.

Blei, D.M., Ng, A.Y. & Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research,* 3 (2003), 993-1022 (blei03a.pdf)

Blei, D.M. (2009). *Topic Models* (mlss09uk_blei_tm.pdf)

Blei, D.M. & Lafferty, J.D. (2009). *Topic Models* (BleiLafferty2009.pdf)

Blei, D.M. (2012). *Probabilistic Topic Models* (MLSS-2012-Blei-Probabilistic-Topic-Models.pdf)

Blei, D.M., Kucukelbir, A & McAuliffe J.D. (2018). *Variational Inference: A Review for Statisticians*, arXiv:1601.00670v9 (1601.00670.pdf)

Canini, K., Shi, L., Griffiths, T. (2009). Online Inference of Topics with Latent Dirichlet Allocation. *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, PMLR 5:65-72, 2009 (canini09a.pdf)

Cardon, D. (2015). *A quoi rêvent les algorithmes. Nos vies à l'heure des big data*. Paris : Seuil, la république des idées

Chang, J. (2015). *Collapsed Gibbs Sampling Methods for Topic Models* (package « lda »). (lda.pdf)

Chen, F., Du, J. & Vuyyuru, M. R. (2018). *Probabilistic and Bayesian Principal Component Analysis* (AM_205_Final_Report.pdf)

Chien, J.-T. & Wu, M.-S. (2008). Adaptive Bayesian Latent Semantic Analysis. *IEEE Transactions on Audio, Speech and Language Processing*. Vol. 16, No 1, January 2008, 198-207 (IEEETrans.ASLP_ABLSA.pdf)

Chien, J.-T., Wu, M.-S & Wu, C-S. (2005). Bayesian Learning for Latent Semantic Analysis. *InterSpeech 2005*, September, Lisbon (i05_0025.pdf)

Présentation par Hsuan-Sheng Chiu (20051013_mistq_Bayesian Learning for Latent Semantic Analysis.pdf)

Chuang, J., Manning, C.D. & Heer, J. (2012). Termite: Visualization Techniques for Assessing Textual Topic Models. *AVI '12, May 21-25, 2012, Capri Island, Italy* (2012-Termite-AVI.pdf)

Coccaro, N. & Jurafsky, D. (1998). Towards better integration of semantic predictors in statistical language modeling. In *Proceedings of ICSLP 98*, Sydney, Australia (Coccaro.pdf)

Delichère, M. & Memmi, D. (2002). *Analyse Factorielle Neuronale pour Documents Textuels*. TALN 2002, Nancy, 24-27 juin 2002 (TALN02.pdf)

Dempster, A.P., Laird, N.M. & Rubin, D.B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*. Series B (Methodological), Vol. 39, No. 1. (1977), pp. 1-38 (DLR77.pdf)

Dessus, P., Lemaire, B. (1999). APex, un système d'aide à la préparation d'examens. *Sciences et Techniques Éducatives*, 6-2, 409-415. (Apex-STE.pdf)

Dhillon, I. S. (2001). *Co-clustering documents and words using Bipartite Spectral Graph Partitioning.* Austin (Texas) : Department of Computer Sciences University of Texas (dhillon2001.pdf)

Dhillon, I. S., Mallela, S., Modha, D. S. (2003). Information-theoretic Co-clustering. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 89–98. ACM. (kdd_cocluster.pdf)

Ducrocq, E. (2014). *Impact des l'hyper-paramètre alpha sur l'algorithme d'analyse de textes latent Dirichlet allocation*. Département de génie informatique. Ecole polytechnique de Montréal (2014_EmileDucrocq.pdf)

Duda, R.O., Hart, P. E. & Stork, D. G. (2001). *Pattern classification*. New York : John Wiley & Sons, Inc. (1e édition 1973)

Fahad, A., Alshatri, N., Tari, Z., Alambri, A., Khalil, I., Zomaya, A.Y., Foufou, S. & Bouras, A. (2014). A Survey of Clustering Algorithms for Big Data: Taxonomy and Empirical Analysis. *IEEE Transactions on Emerging Topics in Computing*, 3 (2). (06832486.pdf)

Fei-Fei, L. & Perona, P. (2005). A Bayesian Hierarchical Model for Learning Natural Scene Categories. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Fei-FeiPerona2005, 10.1.1.86.7111.pdf)

Freitas, N. & Barnard, K. (2001). *Bayesian latent semantic analysis of multimedia databases*. Tech. Rep. Univ. British Columbia, Vancouver, BC, Canada, 2001, TR-2001-15. (nando-blsa.pdf)

Gaussier, E. & Yvon, F. (2011). *Modèles statistiques pour l'accès à l'information textuelle.* HERMÈS/LAVOISIER (https://hal.archives-ouvertes.fr/hal-00742830)

Gefen, D. Endicott, J. E., Fresneda, J. E., Miller, J. & Larsen, K. R. (2017). A Guide to Text Analysis with Latent Semantic Analysis in R with Annotated Code: Studying Online Reviews and the Stack Exchange Community. *Communications of the Association for Information System*s: Vol. 41 , Article 21. (GuideLSI.pdf)

Gimpel, K. (2006). *Modeling Topics* (gimpel.06.pdf)

Gordon, S. & Gilles Bélanger, G. (1996). Échantillonnage de Gibbs et autres applications économétriques des chaînes markoviennes. *L'Actualité économique*, Volume 72, numéro 1, mars 1996 (602194ar.pdf)

Grantham, N. S. (s.d.). *Clustering Binary Data with Bernoulli Mixture Models* (clustering-binary-data.pdf)

Griffiths, T.M. & Steyvers, M. (2004). *Finding scientific topics*. PNAS, vol. 101, 5228-5235 (griffith.pdf)

Griffiths, T. L., Steyvers, M., Blei, D. M., & Tenenbaum, J. B. (2005). Integrating topics and syntax. In *Advances in Neural Information Processing* 17. Cambridge, MA: MIT Press. (2587-integrating-topics-and-syntax.pdf)

Grün, B. & Hornik, H. (2011). topicmodels: An R Package for Fitting Topic Models. *Journal of Statistical Software*, 40(13), 1–30 (v40i13.pdf)

https://www.rdocumentation.org/packages/topicmodels/versions/0.2-8/topics/LDA

Grün, B. & Hornik, H. (2018). *Package 'topicmodels'* (topicmodels.pdf)

Hoffman, M., Blei, D. M. & Bach F. (2010). Online inference for latent Drichlet allocation. *Neural Information Processing Systems*, 2010 (3902-online-learning-for-latent-dirichlet-allocation.pdf)

Hofmann, T. (1999). Probabilistic Latent Semantic Analysis. In *Uncertainity in Artificial Intelligence, UAI'99, Stockholm,* 289-296. (1301.6705.pdf, Hofmann-UAI99.pdf)

Hofmann, T. (2017). Probabilistic latent semantic indexing. *ACM SIGIR Forum*, Vol. 51, N° 2, 211-217. (p211.pdf, 2017)

Khribi, L. (2007). *L'échantillonnage de Gibbs pour l'estimation bayésienne dans l'analyse de survie*. Université du Québec à Montréal (M9739.pdf)

Malisiewicz, T. (2006). *Graphical Models: Recent Trend in Machine Learning* (Bayesian-Hierarchical-Model-for-Learning-Natural-Scenes.pdf)

Maurin, M. (1984). Tests statistiques sur les distributions de Dirichlet. *Statistique et analyse des données*, tome 9, no 3 (1984), p. 45-74 (SAD_1984__9_3_45_0.pdf)

Mazarura, J. & Waal (de), A. (2016). A comparison of the performance of latent Dirichlet allocation and the Dirichlet multinomial mixture model on short text. *IEEE* (PRASA_2016_Mazarurapdf.pdf)

Misra, H., Cappe, O. & Yvon, F. (2015). Using LDA to detect semantically incoherent documents. *WSDM'15*, February 2–6, 2015, Shanghai, China. (W08-2106.pdf)

Murphy, K.L. (2012). *Machine Learning: A Probabilistic Perspective*. Cambridge, MA : The MIT Press (Murphy_ 2012-08-24.pdf)

Obozinski, G. (2012). *LSI, pLSI, LDA and inference methods* (présentation ppt) (pgmir3.pdf)

Popescul, A., Ungar, L.H., Pennock, D.M. & Lawrence, S. (2001). Probabilistic Models for Unified Collaborative and Content-Based Recommendation in Sparse-Data Environments. *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence 2001* (UAI 2001), pages 437-444. (Probabilistic_Models_for_Unified_Collaborative_and.pdf)

Plevoets, K. (2016). *Package « svs », Tools for Semantic Vector Spaces* (svs.pdf)

Pochon, L.-O. & Favre, A. (2007). *Connaissance, théorie de l'information et hypertextes : histoire d'une lecture sélective.* Neuchâtel : IRDP (07.1001).

https://www.irdp.ch/institut/connaissance-theorie-information-hypertextes-853.html

Pochon, L.-O. (2006). *De la possibilité d'usage d'ontologies pour la gestion de contenus mathématiques* : version 04.05. Neuchâtel : IRDP (Document de travail 06.1007)

Pochon, L.-O. (2007). *Vers un modèle formel de classification de problèmes mathématiques et son usage dans la définition de compétences mathématiques*. Neuchâtel : IRDP (Document de travail 07.1002)

Pochon, L.-O. (2019). Problèmes mathématiques sur Internet : l'offre, la manière et l'usage. *La Gazette de Transalpie*, n° 9, octobre 2019, 7-26.

http://www.projet-ermitage.org/doc/Gazzette_transalpie_Pochon_9-2019.pdf

Ponweiser, M. (2012). *Latent Dirichlet Allocation in R*. Institute for Statistics and Mathematics, Wirtschaftsuniversität Wien (thèse de diplôme). (LDA-R.pdf)

Postman, N. (1993). *Technopoly*. New York :Vintage Books.

Prelic, A., Bleuler, S., Zimmermann, P., Wil, A., Bühlmann, P., Gruissem, W., Henning, L., Thiele, L. & Zitzler, E. (2006). A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9),1122–1129.

https://academic.oup.com/bioinformatics/article/22/9/1122/200492 (btl060.pdf)

Röder, M., Both, A. & Hinneburg, A. (2015). Exploring the Space of Topic Coherence Measures. *WSDM'15*, February 2–6, 2015, Shanghai, China. (public.pdf)

Role, F., Morbieu, S., Nadif, M. (2018). *CoClust: A Python Package for Co-clustering*. hal-01804528 (cclust.pdf)

Role, F., Morbieu, S., Nadif, M. (2019). *CoClust Documentation, Release 0.2.1* (coclust.pdf)

Role, F., Morbieu, S., Nadif, M. (2019). CoClust: A Python Package for Co-clustering. *Journal of Statistical Software* 88 (7), 1-29

Rosen-Zvi, M., Griffiths T., Steyvers, M., & Smyth, P. (2004). The Author-Topic Model for Authors and Documents. In *20th Conference on Uncertainty in Artificial Intelligence*. Banff, Canada (The_Author-Topic_Model_for_Authors_and_Documents.pdf)

Sagot Benoît et Fišer Darja (2008). Building a free French wordnet from multilingual resources. In *Ontolex 2008*, Marrakech, Maroc

Santos, F. (2015). *L'algorithme EM : une courte présentation* (algo-em.pdf)

Selivanov, D., Bickel, M. & Wang, Q. (2020). *Package 'text2vec'*

Shashanka,M., Raj, B. & Smaragdi, P. (2008). Probabilistic Latent Variable Models as Nonnegative Factorizations. *Hindawi Publishing Corporation Computational Intelligence and Neuroscience*. Volume 2008, Article ID 947438 (CIN2008-947438.pdf)

Shu, L., Long, B. & Meng, W (2009). *A Latent Topic Model for Complete Entity Resolution*. Conference Paper (https://www.researchgate.net/publication/220967634 ;

A_Latent_Topic_Model_for_Complete_Entity_Resolutio.pdf)

Sievert, C. & Shirley, K.E. (2014). LDAvis: A method for visualizing and interpreting topics. *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces*, pp. 63–70, Baltimore, Maryland, USA, June 27, 2014 (https://github.com/cpsievert/LDAvis, sievert-illivi2014.pdf, W14-3110.pdf)

Sivic, J., Russell, B., Efros, A., Zisserman, A. & Freeman, B. (2005). *Discovering objects and their location in images* (sivic05b.pdf)

Steyvers, M. & Griffiths, T. (2007). Probabilistic Topic Models. In T. Landauer, D. McNamara, S. Dennis, and W. Kintsch (eds), *Latent Semantic Analysis: A Road to Meaning*. Laurence Erlbaum. (SteyversGriffiths.pdf)

Stevens, K, Kegelmeyer, P., Andrzejewski, D., Buttler, D. (2012). Exploring Topic Coherence over many models and many topics. *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Jeju Island, Korea, 12–14 July 2012. pages 952–961 (D12-1087.pdf)

Syed, S. & Spruit, M. (s.d.). *Selecting Priors for Latent Dirichlet Allocation* (Selecting-Priors-for-Latent-Dirichlet-Allocation.pdf)

Tipping, M.E. & Bishop, M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society*. Serie B (Methodological), vol 61, n° 3 (1999), pp. 611-622 (PPCA-KPCA-Notes.pdf)

Wallach, H.M. (s.d.). *Topic models: priors, stop words and language*s (priors.pdf)

Wallach, H.M., Mimno, D. & McCallum, A. (s.d.). *Rethinking LDA: Why Priors Matter*. Department of Computer Science University of Massachusetts Amherst (3854-rethinking-lda-why-priors-matter.pdf)

Wild, F. (2015). *Latent Semantic Analysis* (package « lsa »). (lsa.pdf)

https://www.rdocumentation.org/packages/lsa/versions/0.73.1/topics/lsa

Yildirim, I. (2012). *Bayesian Inference: Gibbs Sampling*. Department of Brain and Cognitive Sciences University of Rochester Rochester (GibbsSampling.pdf)

Zhai, K. & Boyd-Graber, J. (2013). Online Latent Dirichlet Allocation with Infinite Vocabulary *Proceedings of the 30th International Conference on Machine Learning*, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28 (zhai.pdf)

Webographie[1]

https://en.wikipedia.org/wiki/Latent_variable_model

https://en.wikipedia.org/wiki/Probabilistic_latent_semantic_analysis

https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation

https://en.wikipedia.org/wiki/Dirichlet_distribution

https://en.wikipedia.org/wiki/Linear_discriminant_analysis

https://en.wikipedia.org/wiki/Perplexity

https://fr.wikipedia.org/wiki/Classification_na%C3%AFve_bay%C3%A9sienne

https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm

bmabey (2018). I*nteractive topic model visualization. Port of the R package.*

https://pypi.org/project/pyLDAvis/

Benoit, K., Watanabe, K. & al (2013). *Quanteda.textmodels: Scaling Models and Classifiers for Textual Data*

https://cran.r-project.org/web/packages/quanteda.textmodels/index.html

Brooks, A. (2015). *Latent Dirichlet Allocation - under the hood*

http://brooksandrew.github.io/simpleblog/articles/latent-dirichlet-allocation-under-the-hood/

Carlson, J. *Training and Tuning an SVC: R vs Python*

https://joelcarlson.github.io/2016/05/14/RvsPython-GridSearch/

Doll, T. (2018). *LDA Topic Modeling: An Explanation*

https://towardsdatascience.com/lda-topic-modeling-an-explanation-e184c90aadcd

Forum : *Classifying new text using LDA in R* (how can I apply the model on a new batch of documents to classify them among the topics discovered so far?)

https://stackoverflow.com/questions/48268570/classifying-new-text-using-lda-in-r

Evert, S. (2018). *Corpora: Statistics and Data Sets for Corpus Frequency Data*

https://cran.r-project.org/web/packages/corpora/index.html

Jones, T.W. (2019). *Topic modeling*

https://cran.r-project.org/web/packages/textmineR/vignettes/c_topic_modeling.html

Kapadia, S. (2019). *Evaluate Topic Models: Latent Dirichlet Allocation (LDA) ; A step-by-step guide to building interpretable topic models*

https://towardsdatascience.com/evaluate-topic-model-in-python-latent-dirichlet-allocation-lda-7d57484bb5d0

Karani, D. (2018). *Topic modelling with PLSA*

https://towardsdatascience.com/topic-modelling-with-plsa-728b92043f41

Kelechava, M. (2019). *Using LDA Topic Models as a Classification Model Input*

https://towardsdatascience.com/unsupervised-nlp-topic-models-as-a-supervised-learning-input-cf8ee9e5cf28

1 Tous les documents ont été consultés durant l'année 2019.

Konrad, M. *Creating a sparse Document Term Matrix for Topic Modeling via LDA* (procédure R pour créer une matrice DTM utilisée avec Python)

https://datascience.blog.wzb.eu/2016/06/17/creating-a-sparse-document-term-matrix-for-topic-modeling-via-lda/

Kumar, A. *Topic Modelling (Part 1): Creating Article Corpus from Simple Wikipedia dump*

https://appliedmachinelearning.blog/2017/08/28/topic-modelling-part-1-creating-article-corpus-from-simple-wikipedia-dump/

Kumar, A. *Topic Modelling (Part 2): Discovering Topics from Articles with Latent Dirichlet Allocation*

https://appliedmachinelearning.blog/2017/09/28/topic-modelling-part-2-discovering-topics-from-articles-with-latent-dirichlet-allocation/

Kumar, A. *Topic Modelling (Part 3): Document Clustering, Exploration & Theme Extraction from SimpleWiki Articles*

https://appliedmachinelearning.blog/2017/10/13/topic-modelling-part-3-document-clustering-exploration-theme-extraction-from-simplewiki-articles/

Kumar, A. (2018). *Evaluation of Topic Modeling: Topic Coherence*

https://datascienceplus.com/evaluation-of-topic-modeling-topic-coherence/

Li, S. (2016). *Topic Modeling and Latent Dirichlet Allocation (LDA) in Python*

https://towardsdatascience.com/topic-modeling-and-latent-dirichlet-allocation-in-python-9bf156893c24

Mailing list of Gensim, topic modelling for humans :

https://groups.google.com/forum/?hl=amyoxlbezyjbk#!forum/gensim

Mattia's blog, *Ho to do a simple SVM classification in R and Python*

http://mattiacinelli.com/ho-to-do-a-simple-svm-classification-in-r-and-python/

Narayanaswamy, H. (2019). *How to measure topic coherence*

https://labs.imaginea.com/how-to-measure-topic-coherence/

Nie, J.-Y. (s.d). *Information retrieval LSI, pLSI, LDA* (présentation ppt) (LSI-pLSI-LDA.pdf)

Python framework for fast Vector Space Modelling

https://pypi.org/project/gensim-bz2-nsml/

Rul (Van den), C. (2019). *How to Generate Word Clouds in R*

https://towardsdatascience.com/create-a-word-cloud-with-r-bde3e7422e8a

Seidenari, L. (s.d.). Topic models

http://www.micc.unifi.it/seidenari/wp-content/uploads/2010/01/A47-PLSA-Generative-classifier1.pdf (A47-PLSA-Generative-classifier1.pdf)

Tang, F. (2019). *Beginner's Guide to LDA Topic Modelling with R ; Identifying topics within unstructured text*

https://towardsdatascience.com/beginners-guide-to-lda-topic-modelling-with-r-e57a5a8e7a25

Tomar, A. (2018) .*Topic modeling using Latent Dirichlet Allocation(LDA) and Gibbs Sampling*

explained

https://medium.com/analytics-vidhya/topic-modeling-using-lda-and-gibbs-sampling-explained-49d49b3d1045

Wang,H. *Example: Latent Semantic Analysis (LSA)*

https://quanteda.io/articles/pkgdown/examples/lsa.html

Xu, J. *Topic Modeling with LSA, PLSA, LDA & lda2Vec* (Joyce-Xu.pdf)

https://medium.com/nanonets/topic-modeling-with-lsa-psla-lda-and-lda2vec-555ff65b0b05

Yonnet, P. (2012). LDA & LSI : à quoi peuvent-ils servir dans un moteur de recherche

https://fr.slideshare.net/cariboo/lda-lsi

ZhikaiZhang, (2016). *A python implementation of Probabilistic Latent Semantic Analysis using EM algorithm*.

https://github.com/laserwave/plsa

Latent Dirichlet Allocation (LDA) et Google

https://www.scriptol.fr/seo/lda.php

Page de D. M. Blei

http://www.cs.columbia.edu/~blei/

Annexes

Annexe 1 : Parsing et chunking

Plutôt que l'utilisation d'un *parser* basé sur une grammaire, une alternative est souvent utilisée, le « chunking » qui procède à une analyse partiel de la structure de la phrase. Cette analyse peut être basée sur des exemples que fournissent de larges corpus de références. En anglais le plus connu est *Wordnet*[1]. Un logiciel pouvant s'utiliser pour plusieurs langues est *TreeTagger*[2] dont une nouvelle version, *RNNTagger* est implémentée en Python utilisant l'apprentissage profond fourni par la librairie *PyTorch*.

En français de tels corpus sont proposés par le laboratoire de linguistique formelle de l'université Paris-Diderot[3]. Un autre corpus est fourni par le Centre national de ressources textuelles et lexicales[4].

Le projet WOLF (Wordnet Libre du Français)[5] développé en Perl de l'INRIA (Sagot et Fišer, 2008) est un autre projet dont on trouve une description sur la page d'un des administrateurs du projet[6]. En particulier il est précisé que la première version du WOLF (0.1.4) a été construite à partir du « *Princeton WordNet -PWN* » et de diverses ressources multilingues. Le wordnet obtenu a été évalué par rapport au wordnet français issu du projet EuroWordNet. Depuis, des travaux ont encore permis de réaliser des nouvelles versions dont la version 1.0b4 de 2012 qui semble être la version de l'époque de cette chronique. Elle est constituée d'un fichier XML utilisable (wolf-1.0b4.xml) avec l'éditeur DebVisDic[7]. WOLF est une ressource libre distribuée sous la licence Cecill-C licence (LGPL compatible).

Annexe 2 : tf-idf (term frequency–inverse document frequency)

Dans les corpus de texte, il est souvent utile de supprimer les mots outil. Mais d'autres mots peuvent aussi venir grossir inutilement les données à analyser. Ce sont tout d'abord les mots qui ne discriminent pas les documents entre eux et ceux trop rare dans un document.

Parmi les coefficient possibles nous avons utilisé pour le mot *m* du document *d* la valeur

$tfidf(m,d) = fd(m) \times \ln(nd/nm)$ (A2.1) avec $fd(m)$ fréquence de m dans d ; nd nombre de documents et nm nombre de documents comprenant le mot m.

Si tous les documents d'un corpus contiennent le mot m : $tfidf(m,d) = 0$, le mot est sans valeur discriminante. Pour un mot contenu dans un seul document : $tfidf(m,d) = fd(m) \times \ln(nd)$.

Par la suite, on pourra supprimer tous les mots d'un document dont la valeur est inférieure à une valeur de seuil. Certains mots disparaîtront des données.

Un autre usage est de remplacer les fréquences des mots par leur coefficient tf-idf (normalisation).

1 https://wordnet.princeton.edu/

2 https://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/

3 https://recherche.univ-paris-diderot.fr/actualites/french-treebank-un-corpus-de-reference-pour-le-francais dont la demande peut être faire à l'adresse http://ftb.linguist.univ-paris-diderot.fr/

4 https://www.cnrtl.fr/lexiques/morphalou/

5 https://gforge.inria.fr/projects/wolf/

6 http://pauillac.inria.fr/~sagot/index.html#wolf

7 https://deb.fi.muni.cz/proj_debvisdic.php

Avec la définition précédente le coefficient d'un mot présent partout est 0. C'est le cas en particulier lorsque le corpus se compose d'un seul document. Dans ce cas on voudrait que le coefficient soit la fréquence relative du mot dans le document). La formule A2.1 peut être modifiée :

$$tfidf(m,d) = fd(m) \times \ln((1+nd)/nm)$$

si $nd = nm$ on a :

$$tfidf(m,d) = fd(m) \times \ln(1+1/nd) \approx fd(m) \times 1/nd$$

Cette valeur est faible non nulle si le nombre de documents est grand.

Si $nd = 1$: $tfidf(m,d) = fd(m) \times \ln(2)$

Annexe 3 : LSA

Les principales étapes sont les suivantes :

A est la matrice (documents X mots) de dimension M x N.

Par décomposition SVD (unique) : $A = U S V^t$ avec :

- U matrice de dimension M x M orthogonale (i.e $U U^t = I_M$)

- V matrice de dimension N x N orthogonale (i.e $V V^t = I_N$)

- S « diagonale » de dimension M x N avec les valeurs décroissantes.

Ensuite on « coupe » S en une matrice carrée diagonale S_K de dimension K x K et on restreint le nombre de colonnes des matrices U et V :

- U_K matrice (documents X *topics*) de dimension M x K avec $U_K^t U_K = I_K$

- V_K matrice (mots X *topics*) de dimension N x K avec $V_K^t V_K = I_K$

- S_K matrice diagonale d'ordre K

Les colonnes de U_k et V_K restent donc des vecteurs unités orthogonaux.

La matrice $A_K = U_K^t S_K V_K^t = I_K$, également de dimension M x N, est la meilleure approximation de rang K de A selon la norme L2. Les lignes de A_K représentent en quelque sorte des approximations des documents.

Quant à V_K^t , il s'agit de la projection de l'espace des mots dans celui des *topics*. Le produit $V_K^t v$ où v est un vecteur colonne dans l'espace des mots (de dimension M et donc représente un document) donne le vecteur dans l'espace des *topics* (dim K). Cela peut aussi s'écrire $v^t V_K$.

Finalement $D_K = A_k V_k = U_K S_K$ a pour lignes les composantes en *topics* des documents (coordonnées des documents dans l'espace latent). A noter que certaines peuvent se révéler négatives.

On en déduit la formule pour déterminer les coordonnées d'un nouveau document dans l'espace latent.

Pour cela, à partir d'un nouveau document (ou une requête), on détermine le vecteur v donnant les fréquences des mots présents (composantes d'un document dans l'espace des mots). Le produit $T_K = v^t V_K$ en donne les coordonnées dans l'espace latent.

On peut trouver des documents proches en utilisant la distance du cosinus entre T_K et chaque ligne de D_K .

Qu'en est-il de sa relation avec PCA ?

On notera tout d'abord que U est la matrice des vecteurs propres de $B = A A^t$ et $S S^t$ les valeurs propres correspondantes puisque $B = U S V^t V S^t U^t = U S S^t U^t$.

De la même manière, V est la matrice des vecteur propres de B^t .

La différence avec PCA est que cette décomposition ne se fait pas sur A mais sur A centrée (réduite ou non).

A noter que la normalisation tf-idf semble de mise avec LSA ; mais elle est aussi effectuée avec l'analyse PCA par Delichère & Memmi (2002).

Annexe 4 : L'algorithme EM

Le but de l'algorithme itératif EM est de maximiser le coefficient de vraisemblance principalement pour des modèles avec variables latentes. Il est constitué de deux étapes : Estimation et Maximisation. De fait, c'est plutôt un méta-algorithme avec différentes déclinaisons. Dans le cas de pLSA (Hofmann, 1999) il ne s'agit que d'une itération en deux étapes, l'étape de maximisation consistant en un calcul direct.

En faisant une synthèse de diverses présentations générales (Bishop, 2006:439 ; Murphy, 2012:349 ; Santos:4, 2015), il est possible d'en faire une description qui recouvre l'ensemble des cas.

On dispose d'observations multidimensionnelles $\boldsymbol{X} = (X_1, \ldots, X_n)$ qui dépendent d'un paramètre θ (multidimensionnel) de vraisemblance $p(\boldsymbol{X}|\theta)$ qu'il n'est pas possible de maximiser analytiquement.

On considère des variables cachées (latentes, non observées) $\boldsymbol{Z} = (Z_1, \ldots, Z_n)$ indépendantes[1] dont la connaissance permettrait de calculer le log de la vraisemblance des données complètes $\log p(\boldsymbol{X}, \boldsymbol{Z}|\theta)$.

Note : On observe que le calcul des variables cachées et la réduction de dimension sont en quelque sorte un effet de bord de la maximisation de la vraisemblance.

L'estimateur de la log vraisemblance est la moyenne (l'espérance) des probabilités sur chacune des variables cachées $\log p(\boldsymbol{X}, Z_i|\theta)$:

$$\sum_i p(Z_i|\boldsymbol{X}, \theta) \log p(\boldsymbol{X}, Z_i|\theta)$$

Dans l'étape E, on utilise la valeur courante des paramètres θ_{crt} pour trouver la distribution postérieur des variables latentes donnée par $p(Z_i|\boldsymbol{X}, \theta_{crt})$. On utilise alors cette distribution postérieur pour trouver l'espérance (expectation) du log likehood :

$$Q(\theta, \theta_{crt}) = \sum_i p(Z_i|\boldsymbol{X}, \theta_{crt}) \log p(\boldsymbol{X}, Z_i|\theta) \quad \text{(A4.1)}$$

Dans l'étape M on détermine la nouvelle valeur du paramètre en maximisant cette fonction.

$$\theta_{nouv} = argmax_\theta Q(\theta, \theta_{crt})$$

L'algorithme est résumé ci-dessous. Il a la propriété que chaque cycle augmente la valeur du coefficient de vraisemblance (à moins qu'il ait atteint un maximum qui malheureusement peut être local).

1 Par exemple Z_i pourrait être le profil du document X_i en *topics*.

L'algoritme EM

Etant donnée une distribution $p(X, Z|\theta)$ sur des variables observées et des variables latentes Z gouvernées par des paramètres θ, le but est de maximiser le coefficient de vraisemblance $p(X|\theta)$ par rapport à θ.

1. Choix d'une valeur initiale pour les paramètres : θ_{crt}

2. Etape E : évaluer $p(Z|X, \theta_{crt})$.

3. Etape M : déterminer $\theta_{new} = argmax_\theta Q(\theta, \theta_{crt})$ où $Q(\theta, \theta_{crt})$ est donnée par A4.1

4. Vérifier la convergence du log likelihood ou de la valeur des paramètres. Si les critères de convergence ne sont pas satisfaits poser $\theta_{crt} \leftarrow \theta_{nouv}$ et retourner au point 2.

L'algorithme EM peut aussi être utilisé pour trouver le MAP (maximum posterior) pour des modèles où une distribution a priori $p(\theta)$ est définie sur les paramètres. Dans ce cas l'étape E reste identique au cas précédent tandis qu'à l'étape M, la quantité à maximiser est donnée par :

$$\theta_{new} = argmax_\theta (Q(\theta, \theta_{crt}) + \log p(\theta))$$

Un choix judicieux pour la distribution a priori permet de faciliter les calculs.

D'autres techniques sont utilisées pour déterminer les paramètres, notamment l'échantillonnage de Gibbs (Gordon & Bélanger, 1996 ; Anonyme (s.d.) ; Blei, Kucukelbir & McAuliffe, 2018 ; Yildirim,2012 ; Khribi, 2007).

Annexe 5 : Déterministe vs statistique : le cas du polynôme d'interpolation

Bishop (2006) présente la construction d'un modèle probabiliste pour déterminer un polynôme d'interpolation. Il s'agit de calculer les coefficients d'un polynôme de degré *d*

$$f(x, \boldsymbol{w}) = \sum_{i=0}^{d} w_i x^i$$

approchant *N* points $(x_n, y_n = f(x_n))$ en évitant un sur-apprentissage. La première condition est que *d* doit être inférieur à *N*.

Approche déterministe

L'approche déterministe va utiliser la méthode des moindres carrés et minimiser l'erreur :

$$E(\boldsymbol{w}) = \frac{1}{2} \sum_{n=0}^{N} (f(x_n, \boldsymbol{w}) - y_n)^2$$

En dérivant cette expression par rapport à w_i on a :

$$\frac{dE}{dw_i} = \sum_{n=0}^{N} (x_n^i w_i - y_n) x_n^i = 0$$

et donc :

$$w_i^* = \frac{\sum_{n=0}^{N} (x_n^i y_n)}{\sum_{n=0}^{N} x_n^i}$$

Dans ce cas l'erreur des moindres carrés (RMS-error (root-mean-square error)) vaut :

$$E_{RMS}=\sqrt{2E(\boldsymbol{w}^*)/N}$$

Un perfectionnement de la méthode est d'empêcher les coefficients de prendre des valeurs trop grandes (régularisation) et d'utiliser une autre valeur à minimiser :

$$\tilde{E}(\boldsymbol{w})=\frac{1}{2}\sum_{n=0}^{N}(f(x_n,\boldsymbol{w})-y_n)^2+\frac{\lambda}{2}\|\boldsymbol{w}\|^2 \quad \text{(A5.1)}$$

$$\frac{d\tilde{E}}{dw_i}=\sum_{n=0}^{N}(x_n^i w_i-y_n)x_n^i+\lambda w_i=0$$

$$\tilde{w}_i=\frac{\sum_{n=0}^{N}(x_n^i y_n)+\lambda}{\sum_{n=0}^{N}x_n^i}$$

Approche probabiliste

Le modèle probabiliste va introduire une hypothèse sur la distribution d'erreur sur les coefficients, par exemple :

$$p(y_n|x_n,\boldsymbol{w},\beta)=\boldsymbol{N}(y_n|f(x_n,\boldsymbol{w}),\beta^{-1})$$

où $\boldsymbol{N}(x|\mu,\sigma^2)$ est densité de la distribution gaussienne de moyenne μ et de variance σ^2 . $\beta=1/\sigma^2$ donne la précision.

Avec l'hypothèse que les données sont indépendantes, le coefficient de vraisemblance (probabilité d'obtenir les données étant donné les paramètres, $p(D|w)$) à maximiser est (technique du maximum likelihood - ML) :

$$p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w},\beta)=\prod_{n=1}^{N}\boldsymbol{N}(y_n|f(x_n,\boldsymbol{w}),\beta^{-1})$$

Pour effectuer le calcul on passe par le log de la vraisemblance (log likelihood) :

$$\log p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w},\beta)=\sum_{n=1}^{N}\log\boldsymbol{N}(y_n|f(x_n,\boldsymbol{w}),\beta^{-1})$$

$$\log p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w},\beta)=\sum_{n=1}^{N}\log\frac{\sqrt{\beta}}{\sqrt{2\pi}}\exp\{\frac{-\beta}{2}(f(x_n,\boldsymbol{w})-y_n)^2\}$$

$$\log p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w},\beta)=\frac{N}{2}\log\beta-\frac{N}{2}\log(2\pi)-\frac{\beta}{2}\sum_{n=1}^{N}(f(x_n,\boldsymbol{w})-y_n)^2 \quad (1)$$

A un coefficient multiplicatif près la dérivée de l'expression (1) par rapport aux w_i est la même que précédemment. Les coefficients $\boldsymbol{w_{ML}}$ (moyenne) sont donc les mêmes que précédemment avec un bruit gaussien. Quant à la précision β , en maximisant (1) par rapport à β , elle vaut :

$$\frac{1}{\beta_{ML}}=\frac{1}{N}\sum_{n=1}^{N}(f(x_n,w_{ML})-y_n)^2$$

La prédiction *y* pour de nouvelle valeur de x devient :

$$p(y|x,\boldsymbol{w_{ML}},\beta_{ML})=\boldsymbol{N}(y|f(x,\boldsymbol{w_{ML}}),\beta_{ML}^{-1})$$

Modèle Bayésien

L'introduction d'une distribution a priori (prior) sur les coefficients ***w*** constitue un pas de plus vers le modèle bayésien. La probabilité a posteriori est proportionnelle au produit du prior et de la

fonction de vraisemblance :

$$p(\boldsymbol{w}|D)=\frac{p(D|\boldsymbol{w})\,p(\boldsymbol{w})}{p(D)} \quad \text{avec} \quad p(D)=\int p(D|\boldsymbol{w})\,p(\boldsymbol{w})\,d\boldsymbol{w}$$

Ici on prend un prior conjugué :

$$p(\boldsymbol{w}|\alpha)=\boldsymbol{N}(\boldsymbol{w}|\boldsymbol{0},\alpha^{-1}\boldsymbol{I})$$

où α hyper-paramètre (ou méta-paramètre).

Une distribution a priori et a posteriori sont dites conjuguées par rapport à la fonction de vraisemblance si elles sont de ma même famille. La distribution a priori est dite conjuguée a priori (conjugate prior). C'est le cas ici puisque la distribution a posteriori est :

$$p(\boldsymbol{w}|\boldsymbol{x},\boldsymbol{y},\alpha,\beta)\propto p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w},\beta)\,p(\boldsymbol{w}|\alpha)$$

On peut maintenant maximiser la distribution postérieure afin de trouver la valeur la plus probable des $\boldsymbol{w}$ pour le jeu de données (technique du maximum posterior - MAP) . Cela revient à maximiser :

$$\log p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w},\beta)+\log p(\boldsymbol{w}|\alpha)$$

et donc minimiser :

$$\frac{\beta}{2}\sum_{n=1}^{N}(f(x_n,\boldsymbol{w})-y_n)^2+\frac{\alpha}{2}\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w}$$

On retombe sur la formule A5.1 avec $\lambda=\alpha/\beta$.

Le modèle bayésien complet cherche à obtenir la distribution prédite $p(y|x,\boldsymbol{x},\boldsymbol{y},\alpha,\beta)$ (distribution prédictive a posteriori). Les paramètres α et β peuvent être fixés a priori ou inférés à partir des données.

$$p(y|x,\boldsymbol{x},\boldsymbol{y},\alpha,\beta)=\int p(y|x,\boldsymbol{w},\beta)\,p(\boldsymbol{w}|\boldsymbol{x},\boldsymbol{y},\alpha,\beta)\,d\boldsymbol{w}$$

avec

$$p(y|x,\boldsymbol{w},\beta)=\boldsymbol{N}(y|f(x,\boldsymbol{w}),\beta^{-1})$$

et

$$p(\boldsymbol{w}|\boldsymbol{x},\boldsymbol{y},\alpha,\beta)=\frac{p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w},\beta)\,p(\boldsymbol{w}|\alpha)}{\int p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w},\beta)\,p(\boldsymbol{w}|\alpha)\,d\boldsymbol{w}}$$