Apprentissage de co-similarités pour la classification automatique de données monovues et multivues

Clément Grimal

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministérial :

Présentée par

Clément Grimal

Thèse dirigée par **Eric Gaussier**
et codirigée par **Gilles Bisson**

préparée au sein **Laboratoire d'Informatique de Grenoble**
et de **Ecole Doctorale Mathématiques, Informatique, Sciences et Technologies de l'Information**

Apprentissage de co-similarités pour la classification automatique de données monovues et multi-vues

Thèse soutenue publiquement le **11 Octobre 2012**,
devant le jury composé de :

Céline Rouveirol
Professeur, Université Paris Nord, Rapporteur
Gilles Richard
Professeur, Université Paul Sabatier de Toulouse, Rapporteur
Sihem Amer-Yahia
Directrice de Recherche, Université de Grenoble, Examinatrice
Céline Robardet
Maître de Conférence, Université de Lyon, Examinatrice
Eric Gaussier
Professeur, Université de Grenoble, Directeur de thèse
Gilles Bisson
Chargé de Recherche CNRS, Co-Directeur de thèse

Douter de tout ou tout croire sont deux solutions également commodes,
qui l'une et l'autre nous dispensent de réfléchir.
Henri Poincarré

Remerciements

Je tiens à remercier mes directeurs de thèse Éric Gaussier et Gilles Bisson qui m'ont permis de faire cette thèse, et ont su m'accompagner au long de ces trois années. Je souhaite également remercier les membres de mon jury de thèse, en particulier mes rapporteurs Céline Rouveirol et Gilles Richard. La présence de Gilles Richard dans mon jury était d'autant plus importante pour moi qu'il fut mon professeur d'apprentissage automatique en Master 2 Recherche à l'Université Paul Sabatier à Toulouse, en profitant pour me donner le goût de cette discipline.

De plus, je remercie tous les membres de l'équipe AMA avec qui j'ai eu la chance de pouvoir échanger pendant le déroulement de ma thèse. En particulier, je dois chaleureusement remercier l'équipe AMA Juniors, avec qui j'ai passé de très bons moments de détente, mais aussi d'échanges professionnels qui m'ont souvent fait avancer. Plus spécialement : Curi qui m'a permis de toujours travailler dans le calme, Grincheux qui m'a souvent remis dans le droit chemin de la langue française, Koala qui m'a permis d'assumer le bobo-geek qui sommeillait en moi, Alvy qui m'a fait découvrir le tsípouro, et finalement Bourg pour lequel il faudrait que j'ajoute une annexe spéciale tant il m'a soutenu pendant cette aventure.

Je dois également dire un grand merci à tous les membres de ma famille, qui m'ont toujours fait confiance et soutenu dans mes projets professionnels. J'inclus bien sûr mes grands-parents, qui ne sont malheureusement plus tous là, mais qui j'en suis certain auraient été très fiers. Je tiens également à remercier Bernard et Guiloune, qui ont bien souvent apporté un précieux soutien.

Finalement, je souhaite remercier chaleureusement mes amis Astrid, Behrouz, Fanette, Hélène, Julie, Luc, Marion, Perrine, Romain B. et Romain M., toujours là pour me redonner le sourire. Plus particulièrement, un immense merci à Marie-Laure, qui a subi sans broncher mes angoisses et a su les faire s'évanouir et distiller d'estimables conseils.

Et pour terminer, mon ami et colocataire modèle pendant ces années grenobloises Jean, avec qui nous avons pu échanger quotidiennement sur les difficultés de la thèse et qui m'a admirablement bien supporté pendant ces trois années.

Le projet ANR FRAGRANCES

Cette thèse a été financée par le biais du projet ANR FRAGRANCES : Filtrage, Recherche et Annotations dans des Graphes d'Interaction Sociaux. Le projet FRAGRANCES avait pour but de proposer de nouvelles méthodes d'accès à l'information documentaire dans un contexte relationnel et communautaire. Étant techniquement très complexe, le problème nous a demandé d'explorer différentes directions. Nous avons choisi de focaliser les aspects formels et algorithmiques sur les méthodes de la fouille de données qui sont aujourd'hui au cœur du développement de l'accès à l'information.

Les partenaires du projet étaient le centre de recherche européen de Xerox (XRCE), le laboratoire d'Informatique de Paris 6 (LIP6), le moteur de recherche français Exalead et bien sûr le laboratoire d'Informatique de Grenoble (LIG). Dans le cadre de ce projet, plusieurs rapports ont étés rédigés en collaboration entre les partenaires. De plus, nous avons également développés des prototypes logiciels des algorithmes χ-Sim et MVSim présentés dans ce manuscrit. Ces prototypes sont publiquement disponibles et permettent de tester simplement ces méthodes, afin de pouvoir reproduire les résultats publiés.

Le site internet du projet où sont disponibles une présentation détaillée du projet, les rapports produits, les présentations des différents partenaires, ainsi que les prototypes logiciels, est le suivant : `http://fragrances.xrce.xerox.com`.

Table des matières

Liste des figures

Liste des tables

Liste des algorithmes

Introduction

Contexte

Ces dernières années, nous sommes rentrés dans l'âge de l'information, et l'Homme génère des masses de données numériques tous les jours. Le développement rapide d'Internet, et plus généralement des technologies numériques, devenues totalement ubiquitaires avec l'essor des téléphones mobiles, permettent de créer et de partager des quantités d'informations autrefois inimaginables. Si on ajoute à cela la baisse des coûts des supports de stockage comme les disques durs, ainsi que la baisse des coûts de transferts, presque n'importe quel être humain peut accéder à presque n'importe quelle information à n'importe quel instant. Pour illustrer cette forte augmentation des données disponibles, on peut citer le président de Google Eric Schmidt qui disait en 2010, "*entre les débuts de l'humanité et 2003, l'humanité a produit 5 exaoctets [soit 5 milliards de gigaoctets] d'information. Aujourd'hui, nous produisons autant d'information tous les deux jours.*"

Aujourd'hui, le problème n'est plus de pouvoir accéder aux informations, mais de pouvoir se repérer parmi cette masse incroyable de données. Les moteurs de recherche, dont Google est l'exemple emblématique, nous proposent d'accéder à des milliers de pages internet pour quelques mots clés. Pour faire face à ce déluge de données, différents domaines de recherche en informatique, et plus particulièrement en Intelligence Artificielle, se focalisent sur l'apprentissage de motifs réguliers dans les données, la catégorisation automatique d'objets (trouver par exemple le sujet d'un document, parmi un ensemble de sujets connus), ou bien la classification d'objets en groupes homogènes (former des classes de documents similaires).

En effet, malgré l'image commune de l'Intelligence Artificielle qui viserait uniquement à construire des machines les plus intelligentes possibles, les chercheurs de ce domaine ne se concentrent pas tous sur cette tâche particulière. Un des sous-domaines de l'intelligence artificielle est l'apprentissage automatique dont l'objectif est la mise au point de méthodes permettant de trouver des régularités sous-jacentes aux données observées. Ce domaine peut être lui même divisé en plusieurs sous-catégories, suivant le type de tâches qu'il cherche à résoudre :

- l'apprentissage supervisé où le programme apprend un modèle du monde à partir de données connues, afin de pouvoir faire de la prédiction sur de nouvelles données inconnues,

- et l'apprentissage non supervisé où le programme est seulement face à des objets inconnus qu'il doit organiser en groupes homogènes et contrastés.

Il existe bien entendu d'autres catégories de tâches d'apprentissage parmi lesquelles on compte l'apprentissage semi-supervisé, l'apprentissage par renforcement, etc.

Cette thèse va alors se concentrer sur l'apprentissage non supervisé et plus particulièrement l'apprentissage de similarités entre des objets, qui va permettre de les regrouper par la suite en groupes homogènes et contrastés. En effet, la tâche la plus courante en apprentissage non supervisé est appelée classification. Elle consiste à regrouper des objets en classes, de telle façon que les objets d'une même classe soient plus similaires entre eux, qu'ils ne le sont vis-à-vis des objets des autres classes. Etre capable de classifier automatiquement des documents par thème par exemple permet de fournir à un utilisateur une vue plus synthétique des données. En effet, s'il est face à 3000 documents, il sera probablement désemparé s'il cherche une information précise. En revanche, si on classe ces 3000 documents en 10 classes par thème, il sera certainement plus aisé pour lui de s'y retrouver.

Il est d'ailleurs intéressant de remarquer que les êtres humains ont cherché à classifier ou à catégoriser les objets qui les entoure de tout temps. Pour preuve, « Les Catégories » est une des œuvres majeures du philosophe grec Aristote, dans laquelle il définit 10 catégories essentielles qui sont les différentes façons de désigner *ce qui est*, en général. Ce n'est que grâce au récent essor de l'informatique, que nous avons cherché des moyens d'automatiser cette tâche. De plus, même si les chercheurs travaillant sur ce problème sont principalement des informaticiens et mathématiciens, précisons qu'il existe des liens multiples avec d'autres domaines comme les sciences cognitives ou la biologie afin de s'inspirer du vivant pour comprendre ce qu'est *apprendre*. De plus, les domaines d'applications sont également multiples, car la classification peut être utile à l'astronomie ou à l'étude des réseaux sociaux par exemple, à chaque fois que l'on cherche à obtenir une meilleure compréhension des données.

Classiquement, ces objets que l'on cherche à classifier sont décrits par un ensemble d'attributs, que l'on va comparer pour déterminer si deux objets donnés sont similaires ou non. Généralement, ces données sont alors représentées par une matrice dont les lignes sont les objets que l'on cherche à classifier et les colonnes les attributs. Cependant, dans de nombreux cas comme celui d'une matrice documents - mots, les colonnes de cette matrice sont homogènes et représentent simplement un autre type d'objets. On parle alors d'une matrice de relation, car elle décrit la relation entre deux types d'objets : par exemple, les documents et les mots qui les composent, ou bien les gènes et leur expression. Certaines méthodes exploitent alors cette dualité en classifiant simultanément les deux types d'objets concernés : la classification d'un type d'objets fournissant alors une représentation plus synthétique des instances du second type d'objets, qui sont alors plus faciles à classifier, et inversement.

De plus, il est de plus en plus courant de disposer de données que l'on ne peut pas résumer en une seule matrice, mais pour lesquelles il faut plusieurs matrices. On parle alors de données multivues, pour lesquelles chaque vue est représentée par une matrice de relation. Citons l'exemple de films qui peuvent être décrits à la fois par les acteurs qui y jouent, et par des mots-clés les décrivant. Ces données seront alors composées d'une matrice films - acteurs, et

d'une matrice films - mots-clés. Les méthodes capables de prendre en compte toutes les vues des données sont alors assez rares car ce domaine de recherche est plus récent.

Le processus de classification

Afin de positionner au mieux les travaux de cette thèse, nous allons présenter le processus complet de la tâche de classification, tel qu'il se présente dans la plupart des applications. La Fig. 1 présente un schéma de ce processus et de ces 4 étapes. Dans la suite nous allons illustrer ce processus par une tâche de classification automatique de documents.

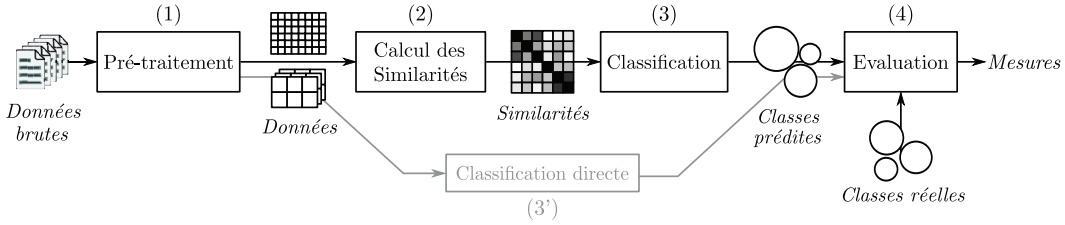


FIGURE 1 – Le processus de classification complet.

L'**étape (1)** consiste à pré-traiter les données brutes, afin de les mettre sous une forme (matrice, graphe, etc.) que les algorithmes utilisés par la suite pourront manipuler. Dans le cas d'une tâche de classification de documents, il s'agira entre autres de lire les documents pour créer la matrice de co-occurrences avec les mots qui les composent. Cette étape inclura certainement des sous-tâches dont le filtrage des mots trop courants et la racinisation [1] des mots.

L'**étape (2)** est le calcul des similarités ou de distances entre les objets que l'on cherche à classifier. Il existe d'ailleurs différentes fonctions permettant de transformer une similarité en distance et inversement. Classiquement, on peut représenter ces valeurs dans une matrice carrée et symétrique.

L'**étape (3)** consiste à créer les classes d'objets à partir de leurs similarités. A cette étape, la méthode de classification utilisée peut soit connaître le nombre de classes des objets, soit découvrir le nombre de classes le plus probable en fonction d'un critère donné (homogénéité des classes découvertes par exemple). Précisons également que certaines méthodes classifient les objets directement à partir des données, et non à partir des similarités, ce que nous avons représenté sur la Fig. 1 par l'**étape (3')**.

Dans une application réelle où un utilisateur cherche à classifier des objets, le processus s'arrête alors ici, et il peut alors analyser la classification obtenue, et éventuellement réaliser une évaluation sémantique grâce à un expert du domaine. Cependant, dans le cas de la mise au point de nouvelles méthodes de classification, des jeux de données pour lesquels on dispose des classes réelles sont utilisés, et l'**étape (4)** d'évaluation consiste alors à comparer les classes prédites aux classes réelles, grâce à des mesures d'évaluation objectives.

1. La racinisation consiste à trouver le radical d'un mot, afin de rassembler les différentes variantes, comme les accords en genre et en nombre par exemple.

Contributions de la thèse

Dans le processus de classification présenté dans la section précédente, les contributions principales de cette thèse se placent dans l'étape (2) de calcul de similarités. En effet, bien qu'il existe des mesures classiques pour la similarité ou la distance entre deux objets, elles ne sont pas toujours adaptées notamment pour traiter le cas où l'on a des collections de matrices de grande taille ou très creuses [2].

La première contribution de cette thèse est l'étude et l'amélioration de l'algorithme de mesure de co-similarité χ-Sim, initialement proposé par Bisson et Hussain (2008), qui calcule la similarité entre toutes les paires d'objets décrits par une matrice de relation. Un nouveau schéma de normalisation est alors proposé, ainsi que l'introduction d'une pseudo-norme et d'une étape de seuillage permettant d'entraver la propagation du bruit dans les similarités calculées. Au delà de ces améliorations, une étude théorique permet une interprétation algébrique de cette approche, et d'autres études sont détaillées.

La seconde contribution est la mise au point de l'algorithme de calcul multivue de co-similarité MVSim qui étend l'algorithme χ-Sim aux données multivues. Cet algorithme permet ainsi de classifier automatiquement des objets décrits, non plus par une seule matrice, mais par plusieurs matrices de relation. De plus, nous proposons une application de cet algorithme pour calculer des similarités entre des objets décrits par une seule matrice de grande taille, pour laquelle l'application directe de l'algorithme χ-Sim ne serait pas possible. Nous décrivons alors un protocole pour « découper » cette matrice en blocs et utiliser MVSim avec ces blocs.

Structure du manuscrit

Le reste de ce manuscrit est organisé en 4 chapitres comme suit :

- Le Chap. I dresse un état de l'art des méthodes de classification, de co-classification et de classification multivue, ainsi que des problèmes liés tels que la représentation des données et l'évaluation des classifications.
- Le Chap. II présente l'étude de l'algorithme χ-Sim, ainsi que les améliorations lui ayant été apportées, et des études théoriques permettant d'illustrer son comportement.
- Le Chap. III décrit notre proposition d'algorithme de calcul multivue de co-similarité, nommé MVSim, et qui peut être vu comme une extension aux données multivues de l'algorithme χ-Sim.
- Le Chap. IV, enfin, détaille finalement les expérimentations ayant été menées afin d'évaluer les performances des algorithmes χ-Sim et MVSim. Sur des jeux de données monovues et multivues (alors utilisés comme plusieurs jeux monovues) pour évaluer χ-Sim, puis uniquement à l'aide des jeux de données multivues pour l'évaluation de MVSim. Des expérimentations permettent également d'illustrer l'application de MVSim à des jeux de données monovues de grande taille.

2. Une matrice creuse est une matrice qui contient une grande proportion d'éléments nuls, en pratique de zéros.

Notations

Nous utiliserons les notations classiques : les matrices seront nommées par des lettres majuscules en caractères gras, les vecteurs par des lettres minuscules en caractères gras, et les variables en caractères italiques.

Nous considérons un ensemble de N **types d'objets** $\mathcal{X} = \{X_n\}_{n=1}^{N}$. Chaque type d'objets est noté X_n, qui comporte η_n instances : $X_n = \{x_i^{(n)}\}_{i=1}^{\eta_n}$.

Ces objets sont décrits par un ensemble de M **vues** $\mathcal{V} = \{V_v\}_{v=1}^{M}$. Chaque vue est notée V_v et plus particulièrement, la notation $V_v : X_n \sim X_{n'}$ signifie que les instances de X_n y sont en relation avec les instances de $X_{n'}$. De plus, cette vue V_v est représentée par une **matrice de relation** $\mathbf{R}_v$, de taille $\eta_n \times \eta_{n'}$. Un élément de $\mathbf{R}_v$ noté $r_{ij}^{(v)}$, décrit l'intensité de la relation entre $x_i^{(n)}$ et $x_j^{(n')}$. Par ailleurs, la ligne i d'une matrice $\mathbf{R}$ sera notée $\mathbf{r}_{i:}$, et la colonne j sera notée $\mathbf{r}_{:j}$.

Pour chaque type d'objets X_n qui doit être classifié, son **nombre de classes réelles** est noté K_n. De plus, la **classification** des instances de X_n est notée $\pi(X_n)$, et elle associe à chaque $x_i^{(n)}$ un entier entre 1 et K_n, ainsi $\pi(x_i^{(n)})$ est la **classe** de $x_i^{(n)}$.

Les similarités entre les instances de X_n sont représentées dans la **matrice de similarité** $\mathbf{S}_n$ dont l'élément $s_{ij}^{(n)}$ est la similarité entre $x_i^{(n)}$ et $x_j^{(n)}$, comprise dans [0,1]. Dans le cas multivue où plusieurs matrices de similarités concernant les instances de X_n seront calculées à partir de différentes vues, on notera la matrice de similarité des instances de X_n calculée à partir de la vue V_v ainsi : $\mathbf{S}_n^{[v]}$. Finalement, on représente cet ensemble de matrices par $\mathcal{S}_n = \{\mathbf{S}_n^{[v]}, \mathbf{S}_n^{[v']}, \ldots\}$.

L'instance i de X_n sera généralement notée $x_i^{(n)}$, mais pourra également être représentée en tant que ligne (ou colonne) d'une matrice de relation $\mathbf{R}$: $r_{i:}$.

Par ailleurs, nous noterons la matrice identité $\mathbf{I}_\eta$ de taille $\eta \times \eta$, ou simplement $\mathbf{I}$ lorsqu'il n'y a pas d'ambiguïté sur sa taille. De plus, nous noterons la transposée d'une matrice $\mathbf{R}$ par $\mathbf{R}^\top$.

Chapitre I

Etat de l'art

Sommaire du chapitre

L'objectif de la classification automatique est d'organiser un ensemble d'objets en groupes homogènes et contrastés, de telle façon que les objets d'un même groupe soient plus similaires entre eux que des objets appartenant à des groupes différents. Cette tâche particulière d'organisation de données est considérée comme non supervisée au sein du domaine plus général de l'apprentissage automatique.

Les notions de distances et de similarités sont souvent au cœur des approches d'apprentissage automatique, en particulier des méthodes de classification, et la plupart des mesures classiques ne sont pas adaptées aux jeux de données réels. En effet lorsque l'on applique ces méthodes à des données réelles, la grande taille de ces données et leur aspect creux rendent le plus souvent ces mesures inappropriées. C'est en partie afin de mieux prendre en compte ces propriétés des données réelles, que des approches de co-classification ont été développées. Ces approches classifient simultanément les multiples types

d'objets décrits par les données, permettant d'obtenir de bonnes performances, même lorsque celles-ci sont très creuses.

Cependant, ces algorithmes de co-classification ne sont adaptés que dans le cas d'une seule vue et de deux types d'objets, et une nouvelle classe de méthodes de classification multivue a alors été développée. Ces approches sont ainsi capables de classifier des objets pour lesquelles les données décrivent de multiples relations avec d'autres types d'objets, par le biais de différentes vues.

Peu importe le type de données, l'objectif final de la classification automatique est de fournir une représentation plus synthétique, plus concise des données, le tout en préservant le plus d'information possible. De nombreuses mesures d'évaluation ont été proposées afin de mesurer la performance de nouveaux algorithmes, se basant le plus souvent sur les classes attendues des objets classifiés.

I-1 Représentation des données

Les données que nous considérerons dans l'ensemble de cette thèse décrivent des relations entre des types d'objets. Ces types d'objets pourront représenter aussi bien des documents, des mots, des films, etc. Et les relations entre ces types d'objets pourront être des relations de co-occurrences, de citations, etc. Une relation peut également concerner les instances d'un seul type d'objets, comme dans le cas des citations entre des articles scientifiques.

Plus formellement, nous supposons que nous avons N types d'objets différents, et nous notons X_i le type d'objets i qui est constitué de η_i instances de ce type d'objets. Décrivant différents types de relations entre ces objets, nous supposons que nous disposons de M vues sur les données, chaque vue V_v représentée par une matrice de relations notée $\mathbf{R}_v$. Nous nous restreignons dans cette thèse, sauf mention contraire, à considérer des relations dyadiques, i.e. concernant uniquement deux types d'objets, mais nous verrons dans la Sec. I-1.1.a comment envisager une extension des méthodes proposées, au cas plus général où une relation peut concerner plus de deux types d'objets. Ainsi, supposons que $V_v : X_i \sim X_j$, alors cette relation décrit l'intensité des liens entre les instances de X_i et les instances de X_j.

La Fig. 2 présente un exemple de jeu de données « jouet » avec 3 types d'objets X_a, X_b et X_c, et 2 relations $\mathbf{R}_1$ liant les instances de X_a à celles de X_b et $\mathbf{R}_2$ liant les instances de X_b à celle de X_c.

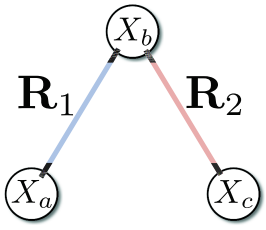


FIGURE 2 – Exemple d'un jeu de données « jouet », avec 3 types d'objets et 2 relations.

Il existe différentes façons de modéliser de telles données, et nous allons voir quelles sont les deux principales dans les sections suivantes : le modèle algébrique qui décrit chaque relation par une matrice, et le modèle à base de graphes. Ces exemples seront également illustrés avec le jeu de données « jouet » de la Fig. 2.

I-1.1 Modèle algébrique

Dans ce modèle classique, chaque relation est représentée par une matrice. $\mathbf{R}_v$ est donc une matrice décrivant la relation entre les objets de types X_i et X_j. Ainsi, la matrice $\mathbf{R}_v$ décrit une relation entre X_i et X_j, est de taille $\eta_i \times \eta_j$, et ces éléments $r_{ab}^{(v)}$ expriment l'intensité du lien entre l'instance a de X_i et l'instance b de X_j. Différents modèles peuvent être utilisés pour décrire cette intensité, les principaux sont détaillés dans le paragraphe sur les modèles de pondération dans la Sec. I-1.1.b.

> ***Remarque 1.*** *Il peut exister plusieurs relations impliquant les mêmes types d'objets. Considérons par exemple que X_i représente des utilisateurs et X_j des films, alors il peut exister une relation exprimant une note attribuée par les utilisateurs sur les films, mais aussi une autre relation indiquant les films que possèdent les utilisateurs, etc. Une relation peut également concerner un seul type d'objets, dans ce cas $i = j$. Un exemple est celui d'articles scientifiques, dont les liens seraient ceux des citations.*

La Fig. 3 représente un jeu de données « jouet » comportant 3 types d'objets : X_a, X_b et X_c ; liés par 2 relations : $\mathbf{R}_1$ liant X_a à X_b et $\mathbf{R}_2$ liant X_b à X_c, pour lesquelles le modèle binaire est utilisé (voir description des modèles de pondération Sec. I-1.1.b).

$\mathbf{R}_1$

X_a \ X_b	$x_1^{(b)}$	$x_2^{(b)}$	$x_3^{(b)}$
$x_1^{(a)}$	1	1	0
$x_2^{(a)}$	1	1	0
$x_3^{(a)}$	0	1	1
$x_4^{(a)}$	0	0	1

$\mathbf{R}_2$

X_b \ X_c	$x_1^{(c)}$	$x_2^{(c)}$	$x_3^{(c)}$	$x_4^{(c)}$	$x_5^{(c)}$
$x_1^{(b)}$	1	1	1	0	0
$x_2^{(b)}$	0	1	1	1	0
$x_3^{(b)}$	0	0	1	0	1

FIGURE 3 – Modèle algébrique pour le jeu de données « jouet », présenté dans la Fig. 2.

I-1.1.a Extension aux relations non-dyadiques

L'extension naturelle de ce modèle au cas des relations polyadiques[3], est couverte par les tenseurs, qui sont eux-mêmes des généralisations des vecteurs et des matrices. En effet, un tenseur d'ordre 1 est un vecteur (1 dimension), un tenseur d'ordre 2 est une matrice (2 dimensions : lignes et colonnes). Pour une relation liant t types d'objets à la fois, on considère alors un tenseur d'ordre t. Un tel exemple de relation se trouve dans les systèmes de notations de films par exemple, où un utilisateur u, va noter la performance d'un acteur a dans un film f avec une note comprise entre 1 et 5. Pour représenter une telle relation, on utilise le tenseur

3. Une relation dyadique est une relation entre deux types d'objet. Une relation polyadique est une relation entre plus de deux types d'objet.

$\mathcal{T}$, qui permet d'associer à tout triplet (u,a,f) un scalaire entre 1 et 5. Il est ensuite possible de projeter un tel tenseur sur 3 matrices, ce qui engendrera cependant une perte d'information :

- une matrice associant à chaque couple (u,a) la moyenne des notes attribuées par l'utilisateur u à l'acteur a,
- une matrice associant à chaque couple (u,f) la moyenne des notes attribuées par l'utilisateur u aux acteurs jouant dans le film f,
- une matrice associant à chaque couple (a,f) la moyenne des notes attribuées à l'acteur a pour sa performance dans le film f.

Relativement peu de travaux utilisent les tenseurs comme modèle de représentation des données (Banerjee *et al.* 2007, Acar et Yener 2009), car la généralisation des méthodes classiques engendrent une complexité souvent trop grande, et également car les relations sont très souvent uniquement dyadiques dans les problèmes réels.

I-1.1.b Modèles de pondération

Comme nous l'avons précédemment vu, les éléments des matrices de relation décrivent l'intensité des liens entre des instances. Par exemple, si $\mathbf{R}_v$ est la matrice de relation entre les objets de type X_i et X_j, alors le terme $r_{ab}^{(v)}$ de cette matrice représente le lien entre l'instance a de X_i et l'instance b de X_j. Supposons que $\mathbf{R}_v$ soit une matrice documents - mots pour simplifier la présentation des différents modèles, en sachant qu'ils restent valides pour d'autres types de données. Nous introduisons la notation suivante : occ_{ab} est le nombre d'occurrences du mot b dans le document a. Les trois approches de pondération des termes que nous considérons dans cette thèse sont les suivants :

- le modèle binaire (absence / présence),
- le nombre d'occurrences,
- et la pondération *TF-IDF* (*Term Frequency - Inverse Document Frequency* en anglais).

Le modèle binaire est le plus simple, et consiste simplement à indiquer si les objets apparaissent ensemble ou pas. Dans le cas du documents a et du mot b, on indique simplement si le mot b est présent dans le document a. Avec ce modèle de pondération :

$$\forall a,b \quad r_{ab}^{(v)} = \begin{cases} 1 & \text{si } \text{occ}_{ab} > 0 \\ 0 & \text{sinon} \end{cases}$$

Si l'on souhaite disposer de plus d'information, on peut choisir de modéliser l'intensité du lien entre deux objets par le nombre de fois où ils apparaissent ensemble. Si le mot b apparaît 5 fois dans le document a, alors le terme $r_{ab}^{(v)}$ vaudra 5. Ce modèle de pondération est simplement :

$$\forall a,b \quad r_{ab}^{(v)} = \text{occ}_{ab}$$

Les deux modèles précédents, bien que très intuitifs souffrent d'un défaut important : ils ne prennent en compte ni la taille des documents, ni la fréquence d'apparition d'un terme dans le corpus. Cela peut ne pas être un problème tant que cela est pris en compte par les mesures utilisées dans la suite. La pondération *TF-IDF* (Salton 1983) est surtout utilisée en recherche

d'information car elle permet de décrire l'intensité du lien entre le mot b et le document a, en considérant l'ensemble du corpus, y compris la taille des documents. En effet, si le terme b apparaît dans tous les documents du corpus, le fait qu'il apparaisse dans le document a ne nous apporte que très peu d'information. Nous définissons maintenant formellement cette pondération, en rappelant que η_i est le nombre de documents dans le corpus, η_j le nombre de mots :

$$\forall a,b \quad r_{ab}^{(v)} = \mathrm{TF}_{ab} \times \mathrm{IDF}_b$$
$$\text{avec} \begin{cases} \mathrm{TF}_{ab} & = \frac{\mathrm{occ}_{ab}}{\sum_{\beta=1}^{\eta_j} \mathrm{occ}_{a\beta}} \\ \mathrm{IDF}_b & = \log\left(\frac{\eta_i}{|\{a, \mathrm{occ}_{ab}>0\}|}\right) \end{cases}$$

où $|\{a, \mathrm{occ}_{ab} > 0\}|$ est le nombre de documents où le mot b apparaît. Le terme *TF* permet d'assurer que l'intensité du lien entre un document et un terme est proportionnel au nombre de fois où ce terme apparaît dans le document, et le terme *IDF* garantie que cette intensité est inversement proportionnelle à la fréquence d'apparition du terme dans l'ensemble du corpus. Le logarithme est utilisé car on sait, empiriquement, que la fréquence d'apparition des termes dans un corpus suit une loi de Zipf (1949), et sans le logarithme le terme *IDF* dominerait ce schéma de pondération en grandissant trop rapidement pour des mots rares.

I-1.2 Graphes multipartis

Dans ce modèle de représentation, chaque instance d'objets est représentée par un sommet d'un graphe multiparti, dans notre cas, le graphe sera plus précisément M-parti. Un graphe $\mathcal{G} = (\mathcal{D}, \mathcal{A})$ est dit M-parti s'il existe une partition de ses sommets $\mathcal{D}$ en M sous-ensembles $\mathcal{D}_1, ..., \mathcal{D}_M$ (que l'on désignera comme les couches du graphe), tels que deux sommets adjacents appartiennent toujours à deux sous-ensembles différents. Ainsi, chaque couche du graphe $\mathcal{D}_i$ correspond à un type d'objets X_i, et donc chaque sommet de $\mathcal{D}_i$ est une instance de X_i. L'ensemble $\mathcal{A}$ des arêtes du graphes contient donc uniquement des arêtes entre des sommets de couches différentes de $\mathcal{G}$, dont les poids représentent l'intensité de la relation entre les instances de ces objets, suivant le modèle de pondération choisi (voir description des différents modèles Sec. I-1.1.b).

> ***Remarque 2.*** *Dans le cas où une relation existerait entre les instances d'un type d'objets donné (comme par exemple pour les citations entre articles scientifiques), le graphe ne serait plus vraiment multiparti, car il existerait des arêtes entre les sommets d'une couche du graphe. Deux solutions sont alors possibles : dupliquer cet couche d'objets, ou remplacer les arêtes par des arcs pour pouvoir prendre en compte des relations non symétriques.*

La Fig. 4 représente le même jeu de données que la Fig. 3 mais en utilisant un graphe 3-parti. Ces deux représentations sont parfaitement équivalentes.

L'extension naturelle de ce modèle au cas des relations non-dyadiques, est couverte par les hypergraphes. La notion d'hypergraphe est une extension des graphes, qui peut également se noter $\mathcal{H} = (\mathcal{D}, \mathcal{A})$ où $\mathcal{D}$ représente les sommets de l'hypergraphe et $\mathcal{A}$ des hyperarêtes. A la

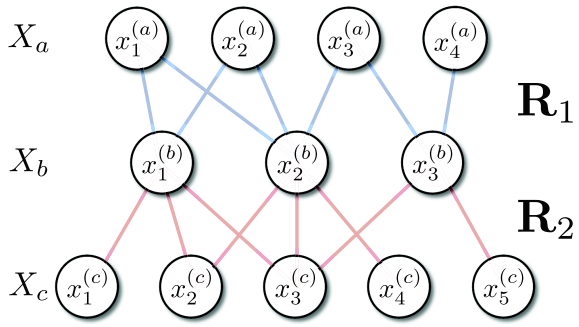


FIGURE 4 – Graphe 3-parti pour le jeu de données « jouet », présenté dans la Fig. 2.

différence d'une arête classique qui lie deux sommets d'un graphe entre eux, une hyperarête lie un ensemble quelconque de sommets entre eux. Pour la relation entre des utilisateurs, des acteurs et des films décrits dans la Sec. I-1.1, on obtient alors des hyperarêtes composée de 3 sommets, liant à chaque fois un sommet de l'ensemble des utilisateurs, un sommet de l'ensemble des acteurs et un sommet de l'ensemble des films. Le poids de cet hyperarête correspondant à la note associée.

I-2 Méthodes de classification et de co-classification

Le problème de la classification automatique a été étudié dans de nombreux contextes différents, et il existe par conséquent de nombreuses approches, dont on peut trouver une étude relativement complète dans Jain *et al.* (1999), Berkhin (2002), Xu *et al.* (2005). L'abondance de recherche dans ce domaine est révélateur de l'intérêt de telles méthodes, permettant d'organiser automatiquement de grands volumes d'informations. Ce type de tâche existe depuis longtemps, mais il existe cependant de nombreux défis encore non relevés.

I-2.1 Distances et similarités

La grande majorité des méthodes de classification (et plus largement des méthodes d'apprentissage automatique) ont besoin d'une notion de distance ou de similarité entre les objets sur lesquels elles travaillent. Plus généralement, certaines méthodes comme la classification hiérarchique nécessite de calculer des distances entres des classes, mais dans cette sous-section, nous nous restreignons aux distances et similarités entre deux objets x_i et x_j, qui sont considérés comme des vecteurs de taille η, et dont la composante l est alors notée x_{il}.

I-2.1.a Les distances

Nous définissons D comme une distance de la façon suivante :

Définition 1. $\forall x_i,x_j,x_k, \mathrm{D}$ *est une distance si elle vérifie les propriétés suivantes :*

1. *Symétrie :* $\mathrm{D}(x_i,x_j) = \mathrm{D}(x_j,x_i)$
2. *Identité des indiscernables :* $\mathrm{D}(x_i,x_j) = 0 \Leftrightarrow x_i = x_j$
3. *Inégalité triangulaire :* $\mathrm{D}(x_i,x_k) \leqslant \mathrm{D}(x_i,x_j) + \mathrm{D}(x_j,x_k)$

De ces 3 propriétés, découle la positivité, à savoir que $\forall x_i,x_j, \mathrm{D}(x_i,x_j) \geqslant 0$.

La distance (en fait la famille de distances) de Minkowski est certainement la plus connue :

$$\mathrm{D}(x_i,x_j) = \left(\sum_{l=1}^{\eta} |x_{il} - x_{jl}|^p \right)^{\frac{1}{p}} \quad (1)$$

Pour $p = 2$, la distance de Minkowski est équivalente à la distance Euclidienne, qui est probablement la plus intuitive (lorsque l'on travaille dans un espace vectoriel du moins) : $\mathrm{D}(x_i,x_j) = \sqrt{\sum_{l=1}^{\eta}(x_{il} - x_{jl})^2}$, qui mesure exactement la *distance* entre 2 points que l'on perçoit dans le plan par exemple. Euclide (né en -325, mort en -265), dans ses travaux fondateurs de la géométrie, démontra que le plus court chemin entre deux points est la droite, ce qui correspond à l'inégalité triangulaire. De plus, pour $p = 1$, la distance de Minkowski devient : $\mathrm{D}(x_i,x_j) = \sum_{l=1}^{\eta} \|x_{il} - x_{jl}\|$, appelé distance de Manhattan, ou distance du taxi car elle correspond à la distance parcouru par un taxi dans une ville américaine où les rues formeraient un quadrillage. Finalement, le cas limite de la distance de Minkowski pour $p = \infty$: $\mathrm{D}(x_i,x_j) = \max_{l=1..\eta} \|x_{il} - x_{jl}\|$ est la distance de Tchebychev.

Pour un calcul de distances entre documents décrits par le nombre d'occurrences de leurs mots, cette famille de distance n'est pas très adaptée. En effet, $(6 - 5) = (1 - 0)$, alors que le fait que deux documents contiennent un même mot respectivement 6 et 5 fois les rend similaires, contrairement au cas où l'un des documents contient un mot (même s'il n'apparaît qu'un fois) et l'autre non. De plus, un autre problème de cette famille de distance est que les caractéristiques avec des valeurs les plus grandes vont dominer les autres. De plus, ces mesures ne prennent pas non plus en compte la taille des objets qu'elles comparent : un long document sera automatiquement plus similaire aux autres de la collection, qu'un document plus court.

Il est également possible de calculer l'entropie relative entre deux distributions d'occurrences de termes, plus connue sous le nom de divergence de Kullback-Leibler, qui mesure le degré de surprise associé à l'observation d'une distribution (le document x_i) alors que l'on en connaissait une autre (le document x_j). Il ne s'agit pas formellement d'une distance, mais d'une dissimilarité, car elle ne vérifie pas la propriété de symétrie et l'inégalité triangulaire. Elle est définie de la façon suivante :

$$\mathrm{KL}(x_i \,||\, x_j) = \sum_{l=1}^{\eta} x_{il} \log \frac{x_{il}}{x_{jl}}$$

Avec $\sum_{l=1}^{\eta} x_{il} = \sum_{l=1}^{\eta} x_{jl} = 1$, car x_i et x_j sont supposées être des distributions de probabilités.

I-2.1.b Les mesures de similarité

Dans le cadre de la classification de documents, à cause des problèmes cités précédemment, on utilise donc plus souvent des mesures de similarités, qui sont définies de la façon suivante :

Définition 2. $\forall x_i,x_j,x_k, \mathrm{S}$ *est une similarité si elle vérifie les propriétés suivantes :*

1. *Symétrie :* $\mathrm{S}(x_i,x_j) = \mathrm{S}(x_j,x_i)$
2. *Positivité :* $0 \leqslant \mathrm{S}(x_i,x_j) \leqslant 1$

3. Identité des indiscernables : $\mathrm{S}(x_i,x_j) = 1 \Leftrightarrow x_i = x_j$

La mesure de similarité la plus souvent usitée dans le contexte de la fouille de texte est la similarité du Cosinus :

$$\mathrm{Cos}(x_i,x_j) = \frac{x_i \cdot x_j}{\sqrt{\|x_i\|_2 \|x_j\|_2}}$$

avec $x_i \cdot x_j$ représentant le produit scalaire entre les vecteur x_i et x_j, et $\|x_i\|_2$ la norme 2 du vecteur x_i. Le numérateur représente donc le cosinus de l'angle θ entre les deux vecteurs, et le dénominateur permet de comparer indifféremment des documents de tailles différentes en normalisant la similarité par le produit de leur norme. Si les documents sont colinéaires, alors leur similarité du cosinus vaudra 1, à l'inverse, s'ils sont orthogonaux (les mots apparaissant dans l'un sont absents de l'autre, et *vice-versa*) alors leur similarité du cosinus vaudra 0. La mesure de similarité du Cosinus peut être utilisée avec le nombre d'occurrences ou la pondération *TF-IDF*. La Fig. 5 illustre la mesure de similarité du Cosinus entre deux vecteurs dans une espace à deux dimensions.

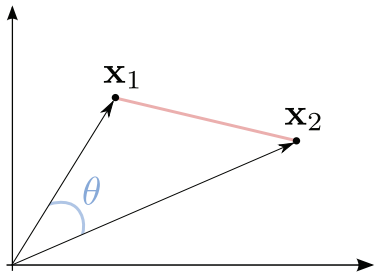


FIGURE 5 – Illustration de la mesure de similarité du Cosinus. Le trait rouge représente la distance Euclidienne, et l'arc figure l'angle θ, dont le cosinus est le numérateur de la mesure.

L'indice de Jaccard (Jaccard 1901), aussi connue sous le nom de similarité de Tanimoto, est le rapport entre la taille de l'intersection et la taille de l'union de deux ensembles. Cette mesure représente une similarité intéressante entre deux documents :

$$\mathrm{Jacc}(x_i,x_j) = \frac{x_i \cdot x_j}{\|x_i\|^2 \|x_j\|^2 - x_i \cdot x_j}$$

Finalement, le coefficient de Dice (Dice 1945), aussi connu sous le nom de similarité de Sørensen, bien qu'initialement prévu pour être appliqué à des données binaires, permet de calculer une mesure de similarité entre deux documents :

$$\mathrm{Dice}(x_i,x_j) = \frac{x_i \cdot x_j}{\|x_i\|^2 \|x_j\|^2}$$

> ***Remarque 3.*** *La mesure* D *induite par la similarité* S *suivante :*
>
> $$\mathrm{D}(x_i,x_j) = \sqrt{2\,(1 - \mathrm{S}(x_i,x_j))}$$
>
> *est une distance valide, vérifiant les 3 propriétés de la Def. 1 (Gower et Legendre 1986).*

I-2.1.c Le fléau de la dimensionnalité

Dans le cadre de l'analyse de corpus de documents textuels, il est admis que les vecteurs représentant les mots et les documents sont très creux, et que ce type de problème souffre par conséquent du fléau de la dimension (ou *curse of dimensionality* en anglais). Ce terme, introduit par Bellman et Kalaba (1959), décrit le fait que dans un espace à très grande dimension (plusieurs centaines ou plusieurs milliers), contrairement à l'espace physique (trois dimensions seulement), le volume de l'espace est si grand que les points qu'on y représente ont tendance à tous devenir équidistants les uns des autres, ce qui pose alors problème pour toutes les méthodes basées sur la distance (ou la similarité) entre ces points. Le nombre de points nécessaires pour obtenir un modèle statistiquement significatif augmente exponentiellement avec le nombre de dimensions de l'espace. Ainsi, les mesures classiques de similarité (ou les distances) ne sont pas adaptées à ces espaces à très grandes dimensions. Dans Aggarwal *et al.* (2001), les auteurs montrent par exemple que dans un tel espace de dimension d où l'on considère une distribution aléatoire de points, si l'on note D_{min} la distance entre les deux points les plus proches et D_{max} la distance entre les points les plus éloignés, alors le ratio $\frac{\mathrm{D}_{max}-\mathrm{D}_{min}}{\mathrm{D}_{min}}$ tend vers 0 quand d tend vers $+\infty$. Intuitivement, ce résultat exprime que quand l'espace devient très grand, les mesures de distance usuelles ne permettent plus de distinguer les points proches des points éloignés.

I-2.2 Méthodes classiques de classification

Il existe de très nombreuses méthodes visant à classifier automatiquement un ensemble de η objets en K classes. Nous ne ferons pas de description exhaustive de tous les types d'approches possibles, et renvoyons le lecteur vers les articles faisant un état des lieux plus complet de ces méthodes (Jain *et al.* 1999, Berkhin 2002, Xu *et al.* 2005). Nous présenterons quelques exemples de méthodes classiques qui seront utilisées dans la suite de ce manuscrit.

I-2.2.a Méthodes de partitionnement

Les méthodes de partitionnement classifient directement les η objets en K classes. Pour un critère donné à optimiser, explorer l'ensemble des partitionnements n'est pas possible, le problème étant NP-difficile, et il faut par conséquent utiliser des heuristiques afin de trouver des solutions approchées. Pour illustrer l'explosion combinatoire de nombre de partitionnements, pour $\eta = 100$ et $K = 10$, il existe plus de 10^{90} possibilités (à comparer au nombre d'atomes dans l'Univers estimés à 10^{80}).

Le critère à optimiser le plus classiquement utilisé est l'erreur quadratique moyenne, qui, intuitivement représente la moyenne des distances entre les points d'une classe, et le point « moyen » de la classe, noté μ :

$$\sum_{i=1}^{K} \sum_{x_j \in c_i} \|x_j - \mu_i\|^2$$

Différentes variantes existent ensuite selon la définition du point « moyen » d'une classe :

- K-moyennes si $\mu_i = \sum_{x_j \in c_i} x_j$, est la moyenne des points de la classe (donc pas nécessairement un objet de la base),

- K-medoïdes si $\mu_i = \underset{x_j^\star \in c_i}{\operatorname{argmin}} \sum_{x_j \in c_i} \|x_j - x_j^\star\|^2$, est le point le plus central de la classe.

Indépendamment de la variante considérée, l'algorithme utilisé pour trouver une solution approchée consiste à initialiser aléatoirement les points « moyens », puis à alterner jusqu'à convergence les deux étapes suivantes :

1. assigner chaque objet au point « moyen » le plus proche,
2. mettre à jour les points « moyens ».

Dans le cas des K-medoïdes, l'algorithme associé sera dans la suite du manuscrit dénoté *PAM* (de l'anglais *Partitioning Around Medoïds*) de Kaufman et Rousseeuw (1990).

Le principal défaut de ces méthodes est la sensibilité à l'initialisation, qui conduit à obtenir des solutions différentes, pour des initialisations différentes, correspondant à des optimums locaux. La version des K-moyennes est également sensible aux points aberrants, alors que la variante des K-medoïdes est plus robuste par rapport à ce critère là. Notons également que ces méthodes nécessitent de connaître le nombre de classes K. L'utilisation de ces méthodes permet d'obtenir des classifications rapidement, même pour de volumineux jeux de données. On peut également noter que des algorithmes ont été proposés afin d'adapter *PAM* à des données de très grandes dimensions : *CLARA* (de l'anglais *CLustering LARge Applications*) de Kaufman et Rousseeuw (1990) et *CLARANS* de Ng et Han (2002).

Finalement, ajoutons qu'il existe de nombreuses généralisations de l'algorithme des K-moyennes, parmi lesquelles on peut citer l'algorithme d'Espérance-Maximisation (Dempster *et al.* 1977), ou EM de l'anglais *Expectation-Maximization*. L'algorithme EM est plus précisément une classe d'algorithmes, qui permet d'apprendre (par maximum de vraisemblance) les paramètres de modèles probabilistes. Pour certaines conditions sur ce modèle probabiliste (distribution normale des données entre autres), on retrouve alors l'algorithme des K-moyennes.

I-2.2.b Classification Hiérarchique

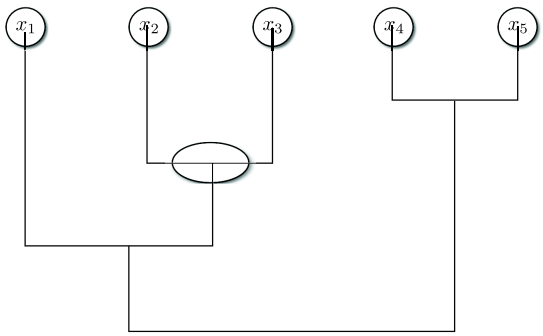
La classification hiérarchique a pour but de fournir une classification plus riche que les méthodes de partitionnement direct, car elle organise les objets en une hiérarchie, classiquement représentée par un dendrogramme. Cette hiérarchie peut être vue comme une emboîtement de partitions des objets. Il existe alors deux types d'approches :

- ascendante : chaque objet est une classe, puis l'algorithme agrège les classes deux à deux ;
- descendante : tous les objets font partie d'une classe unique, qui est divisée au fur et à mesure.

Ces approches supposent donc qu'une mesure de distance (ou de similarité) d est connue entre toutes les paires d'objets. De plus, dans le cas d'une classification ascendante hiérarchique (CAH), on suppose également connue une mesure de dissimilarité entre deux classes, permettant de déterminer quelles classes doivent être fusionnées. Différents critères existent pour calculer cette dissimilarité entre les classes c_1 et c_2 :

- le saut minimum, pour lequel la distance entre deux classes est la plus courte distance entre elles :
$$\mathrm{d}(c_1,c_2) = \min_{x \in c_1, y \in c_2} \mathrm{d}(x,y)$$
- le saut maximum, pour lequel la distance entre deux classes est la plus longue distance entre elles :
$$\mathrm{d}(c_1,c_2) = \max_{x \in c_1, y \in c_2} \mathrm{d}(x,y)$$
- le saut moyen, pour lequel la distance entre deux classes est la distance moyenne entre elles :
$$\mathrm{d}(c_1,c_2) = \frac{\sum_{x \in c_1, y \in c_2} \mathrm{d}(x,y)}{|c_1|\,|c_2|}$$
- le critère de Ward (Ward 1963), pour lequel on minimise la variance intra-classe :
$$\mathrm{d}(c_1,c_2) = \frac{|c_1|\,|c_2|}{|c_1| + |c_2|}\,\mathrm{d}(g_1,g_2)$$
avec g_1 et g_2 les centres de gravité respectifs des classes c_1 et c_2.

Une fois l'algorithme terminé, on dispose donc d'une hiérarchie entre les objets, que l'on représente sous la forme d'un dendrogramme (arbre binaire), dont un exemple est présenté dans la Fig. 6. Finalement, si l'on désire obtenir une classification de ces objets, on peut couper le dendrogramme, à la hauteur fournissant le nombre de classes désiré. Un des avantages de ces méthodes, en plus de fournir une représentation visuelle intéressante, est qu'elles ne nécessitent pas de connaître le nombre de classes *a priori*, mais seulement lorsque l'on veut couper le dendrogramme.



I-2.3 Méthodes de co-classification

Dans de nombreuses applications, les données sont très creuses, c'est à dire que la matrice de relation comporte peu d'éléments non nuls, et les méthodes classiques de classification ne sont pas adaptées à ce genre de données. En effet, la plupart des approches classiques s'appuient sur les caractéristiques communes des objets que l'on veut classifier, et si ces caractéristiques communes sont très rares (les liens entre deux types d'objets), alors la performance de ces méthodes est significativement dégradée. Il a été ainsi remarqué que même dans les cas où l'on s'intéresse uniquement à la classification d'un seul type d'objets , le fait de le co-classifier avec un second type d'objets , permet d'améliorer la classification. Dans le cas de la classification de documents textuels par exemple, où l'on ne s'intéresse pas directement à la classification des mots, les fait de co-classifier les documents et les mots qui les composent simultanément, permet d'obtenir une meilleure classification des documents, qu'une classification directe.

Cette tâche a été intensivement étudiée ces dernières années et ce, dans des domaines différents :

- la recherche d'information, ou plus généralement dans l'étude de corpus de documents textuels (Dhillon 2001, Long *et al.* 2006a, Rege *et al.* 2008, Liu *et al.* 2004, Yen *et al.* 2009, Zanghi *et al.* 2010, Ingaramo *et al.* 2010)
- la bio-informatique, avec pour objectif l'étude de données représentant l'expression de certains gènes (Madeira et Oliveira 2004, Speer *et al.* 2004, Cheng et Church 2000, Iam-on *et al.* 2010)

Dans cette section, nous nous intéressons ainsi aux méthodes de l'état de l'art permettant de traiter le cas où l'on dispose d'une seule vue (donc une seule matrice de relation $\mathbf{R}$), mettant en relation deux types d'objets X_1 et X_2.

I-2.3.a L'analyse sémantique latente (LSA)

Afin de résoudre le problème de la trop grande dimensionnalité des données, dans le cadre de l'analyse de corpus de documents, une des solutions proposée consiste à supposer l'existence de « concepts » (ou *topics* en anglais) latents, liant les documents et les mots qu'ils contiennent. Ce type d'approche est dénommé analyse sémantique latente, ou LSA pour *Latent Semantic Analysis* en anglais (Deerwester *et al.* 1990). Cette méthode a été utilisée avec succès pour différentes applications : recherche d'information (Deerwester *et al.* 1990) et catégorisation de documents (Chakraborti *et al.* 2006) par exemple.

LSA est basé sur la décomposition en valeurs singulières (SVD pour *Singular Value Decomposition* en anglais) de la matrice de données $\mathbf{R}$ représentant le corpus de textes :

$$\mathbf{R} = \mathbf{U} \times \mathbf{\Delta} \times \mathbf{V}^\top$$

avec $\mathbf{U}$ et $\mathbf{V}$ les matrices orthogonales dites respectivement de « sortie » et d'« entrée », et $\mathbf{\Delta}$ la matrice diagonale des valeurs singulières. Cette décomposition permet d'obtenir une approximation de rang inférieur p de la matrice $\mathbf{R}$, en ne gardant que les p plus grandes valeurs singulières. Supposons que les valeurs singulières soient rangées par ordre décroissant

dans la matrice $\mathbf{\Delta}$, et que les matrices de vecteurs singuliers de $\mathbf{U}$ et $\mathbf{V}$ soient permutés en fonction de cet ordre, il suffit alors de ne garder que les p premières colonnes de $\mathbf{U}$, les p premières lignes de $\mathbf{V}^\top$ et les p premières valeurs singulières de $\mathbf{\Delta}$, et de les multiplier afin d'obtenir l'approximation de rang p de la matrice $\mathbf{R}$, comme illustré par la Fig. 7.

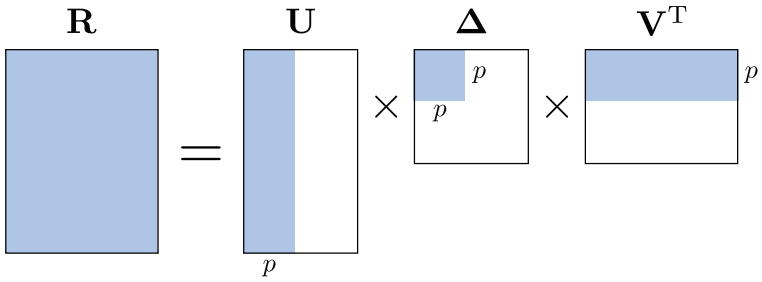


FIGURE 7 – Illustration de la décomposition en valeurs singulières pour l'analyse sémantique latente.

Une fois l'approximation de rang p effectuée, on obtient une projection des données dans un espace de dimension p, dans lequel on peut calculer des similarités, entre deux documents, entre deux termes et également entre un document et un terme, afin de procéder à une classification. Parmi les intérêts de cette approximation de rang p de la matrice $\mathbf{R}$, citons le fait qu'elle est moins creuse que la matrice originale, car les dimensions associées des objets similaires sont fusionnées. De plus, cette opération réduit le bruit présent dans la matrice $\mathbf{R}$ en affaiblissant l'importance des mots qui co-occurrent fréquemment avec de nombreux autres termes. Finalement, LSA capture des co-occurrences d'ordre supérieur à 2 (Kontostathis et Pottenger 2006), c'est à dire, que deux documents ne partageant pas directement des termes, mais des termes similaires, seront similaires dans l'espace projeté de LSA.

Bien que possédant des propriétés intéressantes, la mesure LSA souffre de plusieurs problèmes :

- la décomposition en valeurs singulières de la matrice $\mathbf{R}$ est une opération coûteuse, qui est donc problématique pour des jeux de données de grande taille. Notons tout de même qu'il existe de nombreuses méthodes de calculs itératifs et d'approximations pour résoudre le problème, comme par exemple Brand (2006).
- le paramètre p qui représente le nombre de concepts à garder joue un rôle déterminant dans la qualité de la projection, et dans un contexte non supervisé, il est difficile de le fixer *a priori*.
- la mesure LSA suppose un modèle de distribution Gaussien des données. Cependant, une méthode a été proposée pour résoudre ce problème et généraliser LSA, c'est l'analyse sémantique latente probabiliste PLSA, de l'anglais *Probabilistic Latent Semantic Analysis* (Hofmann 2001).

I-2.3.b Méthodes basées sur la théorie de l'information

Une autre famille de méthodes de co-classification est issue de la théorie de l'information (Slonim et Tishby 2000, Dhillon 2001, Dhillon *et al.* 2003). Ces méthodes s'appuient généralement sur une mesure fondamentale en théorie de l'information, permettant de quan-

tifier la pertinence d'une variable par rapport à une autre, nommée information mutuelle :

$$I(X_1,X_2) = \sum_{x_1,x_2} P(x_1,x_2) \log_2 \frac{P(x_1,x_2)}{P(x_1)\,P(x_2)}$$

Intuitivement, l'information mutuelle permet de mesure combien de *bits* sont nécessaires en moyenne pour transmettre l'information que X_1 contient sur X_2 (et *vice versa*). Elle vaut 0 si et seulement si les variables X_1 et X_2 sont indépendantes.

L'exemple le plus populaire de cette famille d'approche est la co-classification basée sur la théorie de l'information *ITCC*, de l'anglais *Information Theoric Co-Clustering* (Dhillon 2001). Dans ce cas, on suppose que X_1 représente un type d'objets , et X_2 un autre, la co-classification de X_1 et de X_2 doit alors permettre de résumer leur relation, sous la forme de $\hat{X_1}$ et de $\hat{X_2}$, en perdant le moins d'information possible, mesurée par la différence des informations mutuelles : $I(X_1,X_2) - I(\hat{X_1},\hat{X_2})$. Le problème consistant à trouver la classification qui minimise cette différence est NP-complet, mais les auteurs proposent un algorithme permettant de la réduire à chaque itération, permettant d'obtenir au moins un minimum local.

Au delà du problème de minimum local, *ITCC* souffre principalement de deux défauts. Premièrement, en plus de devoir fournir le nombre de classes de documents comme pour beaucoup d'autres méthodes, il faut également fournir à l'algorithme *ITCC* le nombre de classes de mots, alors qu'il est généralement inconnu. Deuxièmement, et à l'instar des k-moyennes, l'initialisation de la méthode est aléatoire, et on obtiendra des performances variables selon cette initialisation. Remarquons que ces deux défauts sont particulièrement problématiques dans un contexte d'exploration des données, où nous n'avons pas la possibilité d'évaluer si une classification est meilleure qu'une autre.

I-2.3.c Méthodes floues

Une des concepts fondamentaux de la logique floue est la fonction d'appartenance, qui modélise l'appartenance d'un point à un ensemble, généralisant ainsi la notion de fonction indicatrice. Ce concept d'appartenance a alors été utilisé pour la classification, mais également adapté pour la co-classification, en particulier concernant des tâches sur des documents et des mots (Oh *et al.* 2001, Kummamuru *et al.* 2003, Tjhi et Chen 2005). En se basant sur une matrice de données $\mathbf{R}$ de taille $\eta_1 \times \eta_2$, le but est de trouver les groupes de lignes et de colonnes les plus « denses ». S'appuyant sur la notion d'appartenance floue, chaque ligne a pourra appartenir à plusieurs classes c, ce qui sera représenté par ses coefficients d'appartenance u_{ca}. Pour une ligne donnée, la somme des ces appartenances aux classes devra être normalisée : $\sum_{c=1}^{K} u_{ca} = 1$. De façon similaire, l'appartenance de la colonne b à la classe c sera représentée par le terme d'appartenance v_{cb}, qui sera normalisé de façon différente : $\sum_{b=1}^{\eta_2} v_{cb} = 1$.

L'objectif est alors de maximiser une fonction objectif, que l'on peut écrire de la façon suivante :

$$\max_{u_{ca},v_{cb}} J = \sum_{c=1}^{K}\sum_{a=1}^{\eta_1}\sum_{b=1}^{\eta_2} u_{ca}\, v_{cb}\, r_{ab} - F$$
$$\text{avec} \sum_{c=1}^{K} u_{ca} = 1$$
$$\sum_{b=1}^{\eta_2} v_{cb} = 1$$

Le premier terme sous forme d'une triple somme est le degré agrégation, et doit idéalement être maximisé. Comme dans la plupart des algorithmes de cette famille, il faut ajouter un terme F – généralement appelé fuzzifier – qui permet d'obtenir des degrés d'appartenance floues et de contrôler le degré de recouvrement des différentes classes. Il existe alors plusieurs variantes pour définir ce fuzzifier :

- Dans l'algorithme *FCCM* (de l'anglais *Fuzzy Clustering for Categorical Multivariate data*) présenté dans Oh *et al.* (2001), basé sur la notion d'entropie floue :

$$F = F_u \sum_{c=1}^{K}\sum_{a=1}^{\eta_1} u_{ca}\,\log(u_{ca}) + F_v \sum_{c=1}^{K}\sum_{b=1}^{\eta_2} v_{cb}\,\log(v_{cb})$$

- Dans l'algorithme *Fuzzy CoDoK* (de l'anglais *Fuzzy Co-clustering of Documents and Keywords*) présenté dans Kummamuru *et al.* (2003), qui vise à régler des problèmes de passage à l'échelle de *FCCM* dûs à des instabilités numériques, basé sur le coefficient de Gini :

$$F = F_u \sum_{c=1}^{K}\sum_{a=1}^{\eta_1} u_{ca}^2 + F_v \sum_{c=1}^{K}\sum_{b=1}^{\eta_2} v_{cb}^2$$

- Dans *FCC-STF* (de l'anglais *Fuzzy Co-Clustering with Single Term Fuzzifier*) présenté dans Tjhi et Chen (2005), où le fuzzifier est composé d'un unique terme :

$$F = -F_{uv} \sum_{c=1}^{K}\sum_{a=1}^{\eta_1}\sum_{b=1}^{\eta_2} \left[(u_{ca} - v_{cb}) - u_{ca}\, v_{cb}\right]^2$$

Les termes F_u, F_v et F_{uv} sont des paramètres permettant de régler le degré de flou, respectivement pour l'appartenance aux classes, des lignes, des colonnes ou des deux simultanément.

Finalement, et ce pour les trois méthodes présentées, les contraintes sont intégrées de façon classique par le biais de multiplicateurs de Lagrange, la fonction objectif devenant ainsi :

$$\max_{u_{ca},v_{cb}} J = \sum_{c=1}^{K}\sum_{a=1}^{\eta_1}\sum_{b=1}^{\eta_2} u_{ca}\, v_{cb}\, r_{ab} - F + \sum_{a=1}^{\eta_1} \lambda_i \left(\sum_{c=1}^{K} u_{ca} - 1\right) + \sum_{c=1}^{K} \gamma_c \left(\sum_{b=1}^{\eta_2} v_{cb} - 1\right)$$

où λ_i et γ_c sont les multiplicateurs de Lagrange.

L'algorithme consiste ensuite à initialiser les appartenances u_{ca} de façon aléatoire, puis à répéter jusqu'à convergence de ces appartenances les mises à jour des v_{cb} et des u_{ca}. Les fonctions de mise à jour ayant été obtenues en fixant les dérivées partielles (par rapport aux u_{ca} et v_{cb}) de J à 0.

De façon classique, pour pouvoir obtenir une classique des données, il faut affecter chaque objet à une classe à partir de ses degrés d'appartenance. Cette étape est appelée defuzzification, et la façon la plus simple de procéder consiste à affecter chaque objet à la classe pour laquelle son degré appartenance était le plus grand.

I-2.3.d Méthodes spectrales

Les méthodes spectrales sont généralement basées sur le partitionnement de graphes (Dhillon 2001, Rege *et al.* 2008, Yen *et al.* 2009, Zanghi *et al.* 2010), et de plus, leur équivalence avec des méthodes de décomposition non-négatives de matrices (Li et Ding 2006, Tang *et al.* 2009, Greene et Cunningham 2009, Arora *et al.* 2011) a été démontrée par Ding *et al.* (2005). Dans le cas de la classification de documents, le graphe utilisé n'est alors pas systématiquement le graphe biparti des documents et des mots, mais fréquemment le graphe projeté des documents. Il est en effet possible de construire un graphe simple de documents à partir du graphe biparti documents - mots. Pour ce faire, on supprime les sommets représentant des mots, et on ajoute une arête entre deux sommets représentant des documents s'ils partageaient des mots (on peut bien sûr pondérer ces arêtes en fonction du nombre de mots partagés, et de leur nombre d'occurrences dans les documents concernés). En considérant le modèle algébrique, cela revient à calculer la matrice d'adjacence du graphe en calculant $\mathbf{A} = \mathbf{R} \times \mathbf{R}^\top$.

Le principe général des méthodes spectrales, est classiquement de trouver la coupe minimale dans le graphe biparti des documents et des mots, ou directement dans le graphe projeté des documents. Pour un graphe $\mathcal{G} = (\mathcal{D}, \mathcal{A})$, on appelle coupe de $\mathcal{G}$ un sous-ensembles d'arêtes $\mathcal{A}'$ qui partitionne les sommets du graphe en deux ensembles $\mathcal{D}_1$ et $\mathcal{D}_2$. La coupe minimale est alors la coupe dont la somme des poids des arêtes de $\mathcal{A}'$ est minimale. On comprend intuitivement que si l'on souhaite trouver deux classes des objets, ce partitionnement est pertinent car c'est celui qui minimise les liens inter-classe. Dans la suite, nous avons choisi de détailler l'une de ces méthodes, car elle nous paraît particulièrement intéressante : la mesure *CTK*.

Le noyau du temps d'aller-retour (de l'anglais *Commute Time Kernel*) a été développé par Yen *et al.* (2009), et cherche à partitionner le graphe projeté des documents. Plus particulièrement, cette méthode se base sur la matrice Laplacienne du graphe qui est obtenue ainsi : $\mathbf{L} = \mathbf{D} - \mathbf{A}$ où $\mathbf{D} = \mathrm{Diag}(a_{i\cdot})$ est la matrice de degrés du graphe, car $a_{i\cdot} = \sum_j a_{ij}$.

L'idée de base derrière cette mesure de similarité est de considérer un marcheur aléatoire se déplaçant sur ce graphe de document. Lorsqu'il se situe sur un sommet, il choisit le sommet suivant en suivant les arêtes du graphe, en suivant un modèle probabiliste très simple : plus le poids de l'arête est fort, plus la probabilité que le marcheur l'emprunte est importante. Ce modèle est très classique et est entre autres utilisé par le moteur de recherche *Google*

pour calculer un score pour chaque page internet en fonction de la quantité de liens entrants. Afin de calculer une mesure de similarité entre deux sommets de ce graphe, l'idée est alors de considérer le temps moyen que mettrait un tel marcheur aléatoire pour faire l'aller-retour entre ces deux sommets.

Pour deux sommets a et b du graphe, la distance du temps d'aller-retour se calcule à partir de la matrice pseudo-inverse de la Laplacienne du graphe $\mathbf{L}^+$:

$$CTK(a,b) = v_G \left(\mathbf{e}_a - \mathbf{e}_b\right) \mathbf{L}^+ \left(\mathbf{e}_b - \mathbf{e}_a\right) \tag{2}$$

avec $v_G = \sum_{i,j} a_{ij}$ désignant le volume du graphe, et $\mathbf{e}_a$ représentant le vecteur de la base canonique dont le terme a vaut 1 et tous les autres valent 0.

Les auteurs montrent également que la matrice $\mathbf{L}^+$ contient les produits scalaires entre les sommets du graphe dans un espace Euclidien où ces sommets sont exactement séparés par la distance du temps d'aller-retour. Ainsi, $\mathbf{L}^+$ est un noyau, et les auteurs, afin de normaliser ses valeurs entre 0 et 1, y appliquent finalement une transformation sigmoïde, qui est une transformation terme-à-terme des éléments de $\mathbf{L}^+$, notés l^+_{ij}, afin d'obtenir le noyau du temps d'aller-retour $\mathbf{K}_{\text{ct}}$:

$$(\mathbf{K}_{\text{ct}})_{ij} = \frac{1}{1 + e^{-\alpha\, l^+_{ij}/\sigma}} \tag{3}$$

avec les paramètres α et σ. Cependant, le paramètre σ correspond à la variance des termes de $\mathbf{L}^+$, et sera donc déterminé automatiquement. Quant au paramètre α, les auteurs le déterminent par des tests préliminaires sur les données. Intuitivement, le rôle de ce paramètre α est de modifier la répartition des valeurs de similarités dans l'intervalle [0,1]. Plus α est grand, plus les valeurs sont soit très proches de 0, soit très proches de 1, avec peu de similarités moyennes.

I-3 Méthodes de classification multivues

A notre connaissance, la notion de multivue dans le contexte de l'apprentissage automatique a été pour la première fois utilisé dans Blum et Mitchell (1998). Dans cet article, les auteurs introduisent la notion de co-apprentissage, ainsi dénommée car elle consiste à entraîner séparément deux algorithmes sur deux vues distinctes, et d'utiliser les prédictions sur des exemples non étiquetés d'un algorithme entraîné sur une vue, pour augmenter l'ensemble d'apprentissage de l'autre algorithme. Par ailleurs, Blum et Mitchell (1998) introduisaient également la notion d'apprentissage semi-supervisé, utilisant à la fois des données étiquetées et non étiquetées pour apprendre un modèle de prédiction.

Les données multivues ont par la suite été également considérées dans le cadre de tâche d'apprentissage non supervisé. D'un point de vue terminologique, on retrouve des problèmes similaires sous plusieurs appellations, parmi lesquelles :

- la classification dans des univers parallèles (Wiswedel et Berthold 2007, Wiswedel *et al.* 2010),

- la classification à partir de différentes sources (Tang *et al.* 2009),
- la classification de multiples variables (Friedman *et al.* 2001, Bekkerman *et al.* 2005),
- la classification de données hétérogènes (Zeng *et al.* 2002, Chen *et al.* 2009),
- la classification de données multitypes (Wang *et al.* 2003, Long *et al.* 2006b, Wang *et al.* 2006).

Le terme de classification multivue, a été utilisé pour la première fois par Bickel et Scheffer (2005), puis a été repris dans de nombreuses publications, et c'est le terme que nous choisissons d'utiliser. Nous allons alors présenter dans cette section, différentes familles de méthodes de classification multivues de l'état de l'art, dans le cas où les données sont donc composées de plusieurs matrices de données $\mathbf{R}_v$, décrivant les relations entre plusieurs types d'objets X_i.

I-3.1 Méthodes floues

Nous allons voir dans cette partie que les méthodes de classification et de co-classification s'appuyant sur la notion de fonction d'appartenance floue (voir Sec. I-2.3.c), ont été adaptées au contexte multivue. On peut notamment citer les travaux de Pedrycz (2002), Frigui *et al.* (2007), Wiswedel et Berthold (2007), Wiswedel *et al.* (2010).

De façon similaire aux méthodes de co-classification, ces méthodes sont basées sur la notion de prototype d'une classe, et sur la définition d'une fonction objectif à optimiser. Un prototype est un objet artificiel, représentant d'une classe, et le terme le plus important de la fonction objectif à minimiser consiste alors en la somme des distances des objets aux prototypes, pondérée par les degrés d'appartenance de ces objets aux classes. Un paramètre permettant de régler le degré de flou est généralement intégré également.

Pedrycz (2002) ajoute à la fonction objectif des termes représentant les degrés de collaboration entre les différentes vues, qui permettent de mesurer si deux vues sont d'accord ou non, et qui sont optimisés simultanément aux degrés d'appartenance des objets aux classes, et aux prototypes.

Une autre approche consiste à associer à chaque objet un degré d'appartenance à chaque vue, en plus d'un degré d'appartenance à chaque classe (Wiswedel et Berthold 2007), ce qui permet de découvrir si un objet est bien représenté dans une vue ou non, en fonction de son intégration à la classification dans cette vue. Un paramètre permet également de contrôler le degré de flou désiré à ce niveau là. Le défaut de ces deux approches est que l'optimisation est effectuée de façon séparée dans chaque vue.

Afin de remédier à ce défaut et à d'autres points faibles de la méthode présentée dans Wiswedel et Berthold (2007), comme la difficulté d'interpréter les résultats (chaque objet ayant un degré d'appartenance à chaque cluster de chaque vue, et un degré d'appartenance à chaque vue), Wiswedel *et al.* (2010) proposent de remplacer le degré d'appartenance d'un objet à une vue, par un degré d'appartenance des classes aux vues. Ceci a plusieurs avantages : permettre d'avoir un seul nombre de classes dans les multiples vues, que les prototypes des classes soient valables dans chaque vue, et que les résultats en deviennent plus facilement interprétables.

Finalement, dans certains cas il n'est pas possible d'avoir accès aux données décrivant les objets, mais en revanche, on dispose de multiples matrices de distances concernant les mêmes objets, ayant été obtenues soient à partir de différentes vues, soient à partir de différentes mesures (Frigui *et al.* 2007). Dans cette approche, nommée *CARD* (de l'anglais *Clustering and Aggregating Relational Data*), il n'est alors plus possible de s'appuyer sur la notion de prototype, et la fonction objectif fait alors uniquement appel aux distances entre des paires d'objets, pondérées par leurs degrés d'appartenance aux classes. De plus, des poids de pertinence sont appris pour chaque paire classe - vue, mesurant si une vue est utile pour la définition d'une classe. Les auteurs introduisent deux variantes de leurs algorithmes, basées sur les deux méthodes de classification floues suivantes : *RFCM* (de l'anglais *Relational Fuzzy C-Means*) de Hathaway *et al.* (1989) et *FANNY* (de l'anglais *Fuzzy ANalYsis clustering*) de Kaufman et Rousseeuw (1990).

Dans toutes ces approches, les contraintes de normalisation sur les différents degrés d'appartenance sont intégrées au problème d'optimisation en utilisant des multiplicateurs de Lagrange, et le problème est finalement résolu grâce aux conditions nécessaires imposant que les dérivées partielles par rapport aux degrés d'appartenance et aux multiplicateurs de Lagrange soient nulles. Ceci permet d'obtenir des règles de mise à jour pour les prototypes s'il y en a, et les différents degrés d'appartenance selon les méthodes. Les algorithmes qui en découlent se résument à initialiser de façon aléatoire les prototypes et degrés d'appartenances, puis à les mettre à jour alternativement jusqu'à ce qu'un critère d'arrêt soit atteint, correspondant généralement à la stabilisation des degrés d'appartenance des objets aux classes. De façon classique, une étape de defuzzification est finalement nécessaire pour affecter chaque objet à une seule classe.

I-3.2 Méthodes issues de la théorie de l'information

Dans la Sec. I-2.3.b, nous avons décrit des méthodes basées sur la théorie de l'information permettant de co-classifier deux types d'objets, et des extensions ont été proposées pour le cadre multivue (Friedman *et al.* 2001, Bekkerman *et al.* 2005). A l'instar des méthodes monovues, ces méthodes s'appuient sur la notion d'information mutuelle qu'elle étend au cadre multivue. Dans ce contexte, les différents types d'objets X_i sont considérés comme des variables aléatoires, et l'information mutuelle qui a été initialement définie pour deux variables X_1 et X_2, est alors étendue à l'ensemble de N variables $X_1, \ldots, X_N$.

Friedman *et al.* (2001) se basent sur les travaux fondateurs en classification issue de la théorie de l'information (Tishby *et al.* 1999), et définissent l'information mutuelle pour de multiples types de variables, appelée *multi-information* :

$$I(X_1, \ldots, X_N) = D\left(P(X_1, \ldots, X_N) \,||\, P(X_1), \ldots, P(X_N)\right)$$

où D est une divergence de Bregman [4], et P la distribution des variables $X_1, \ldots, X_N$. De plus, les relations entre les différentes variables (donc les différents types d'objets) sont représentées

4. Une divergence de Bregman est une classe de mesures généralisant de nombreuses distances classiques (Euclidienne, Kullback-Leibler, etc.), mais qui n'est pas nécessairement symétrique et qui ne vérifie pas nécessairement l'inégalité triangulaire.

par un réseau bayésien, ainsi la distribution de probabilité jointe des variables est contrainte par ce réseau de variables. Cependant, il n'est pas certain que cette distribution jointe ait toujours un sens, car il ne sera pas toujours possible de représenter toute l'information dans une seule distribution (Bekkerman *et al.* 2005).

C'est ainsi qu'une autre extension de la définition d'information mutuelle à de multiples variables est proposée dans Bekkerman *et al.* (2005), sous la forme d'une version factorisée, représentant les interactions entres des paires de variables. La forme de la fonction à maximiser est alors la suivante :

$$\sum_{i,j} w_{ij}\, I\,(X_i,X_j)$$

dans laquelle les w_{ij} sont des poids permettant de pondérer les différentes relations, ces poids étant non nuls si et seulement si $X_i \sim X_j$. Ensuite, ce critère permet de corriger les classifications des différents types d'objets ayant été obtenues grâce à un algorithme de classification hiérarchique, soit ascendante si le nombre de classes est relativement important, soit descendante dans le cas où le nombre de classes est faible, pour des raisons de rapidité d'exécution.

I-3.3 Méthodes probabilistes

Comme dans le cadre de la classification et de la co-classification, des méthodes probabilistes, basées sur la description d'un modèle génératif, ont été développées pour le contexte multivue (Taskar *et al.* 2001, Zeng *et al.* 2002). On retrouve dans ces méthodes la notion de classes latentes, que l'on cherche à retrouver, pour chaque type d'objets, et qui seraient à l'origine de la génération des instances. Dans ces approches, les auteurs cherchent alors à trouver les paramètres de ces distributions de probabilités conditionnelles. Taskar *et al.* (2001) estiment les valeurs des paramètres θ en utilisant la méthode du maximum de vraisemblance, c'est à dire qu'ils cherchent à maximiser $P(D \mid \theta)$ avec D la distribution de probabilité correspondant aux objets observés.

De façon similaire, Zeng *et al.* (2002) cherchent à découvrir les probabilités conditionnelles de la forme $P(k_i^{(n)} \mid k_j^{(n')})$ où $k_i^{(n)}$ est la classe i des X_n. Ainsi si une instance de X_n a une forte probabilité d'appartenir à la classe $k_i^{(n)}$, alors les instances de $X_{n'}$ auxquelles il est lié, auront une probabilité d'appartenir à la classe $k_j^{(n')}$ conditionnée par $P(k_i^{(n)} \mid k_j^{(n')})$. Contrairement à Taskar *et al.* (2001), les auteurs utilisent un algorithme itératif qui alterne entre les vues pour estimer les paramètres de leur modèle. L'algorithme classifie alors tour à tour les instances de chaque type d'objets, modifiant ainsi les liens aux instances des autres types, et donc leur description. L'algorithme s'arrête lorsque les classifications sont stables.

Dans l'analyse de leur approche, Zeng *et al.* (2002) évoquent le fait que les classifications des objets dans les différentes vues se renforcent mutuellement (même si les auteurs ne donnent pas de garantie sur ce point). Partant d'une idée de départ similaire, Wang *et al.* (2003) ont proposé l'algorithme *ReCom* (de l'anglais *Reinforcement Clustering of Multi-type interrelated data objects*), basé sur l'idée que la classification des instances d'un type d'objets, en rendant leur description plus concise, va permettre de rendre les données moins creuses, et donc d'améliorer les classifications des autres types d'objets.

I-3.4 Méthodes spectrales

Le cadre des méthodes de classifications spectrales a aussi été étendu aux problèmes multivues, et il est certainement celui dans lequel on trouve le plus de publications (de Sa 2005, Long *et al.* 2006a, Zhou *et al.* 2007, Kumar et Daume III 2011). Comme dans le cas monovue, ces méthodes se basent sur l'étude sur les vecteurs propres de la matrice Laplacienne du graphe. Cependant, il existe plusieurs façons de définir la matrice Laplacienne, et spécialement dans le contexte multivue, plusieurs graphes pouvant représenter les données.

L'approche proposée dans de Sa (2005) se limite au cas de données composées de deux vues, et construit un graphe biparti dans lequel chaque vue correspond à une couche du graphe. Chaque objet apparaissant dans une vue, est alors représenté par un sommet dans la couche associée à cette vue, et les instances présentes dans les deux vues sont donc représentées par deux sommets. Les auteurs cherchent ensuite comment créer des arêtes entre les sommets des deux couches. Pour cela, ils s'appuient sur les instances communes aux deux vues, et sur la similarité entre ces instances et celles apparaissant uniquement dans l'une des vues. Finalement, les auteurs calculent la classification par un algorithme spectral classique appliqué au graphe biparti ainsi construit.

Dans Kumar et Daume III (2011), les auteurs s'appuient sur l'hypothèse de compatibilité entre les vues, sous-jacent au cadre du co-apprentissage (Blum et Mitchell 1998). Ainsi, si deux points sont dans la même classe en les classifiant dans une vue, alors ils devraient également se retrouver dans la même classe dans les autres vues. Comme la classification peut être une opération coûteuse en temps de calculs, les auteurs proposent une approche ne nécessitant pas d'aller jusqu'à la classification des objets dans chaque vue à chaque étape de l'algorithme. Chaque vue v correspond à une matrice de similarité $\mathbf{S}_v$ (décrivant la structure du graphe de cette vue), à laquelle on peut associer une matrice Laplacienne $\mathbf{L}_v = \mathbf{D}^{-1/2}\,\mathbf{S}_v\,\mathbf{D}^{-1/2}$ où $\mathbf{D}$ est la matrice diagonale dont les termes sont les sommes des lignes de $\mathbf{S}_v$. Les auteurs cherchent ensuite les K plus grands vecteurs propres de la Laplacienne $\mathbf{L}_v$, qui sont les lignes colonnes de la matrice $\mathbf{U}_v$. On peut ainsi utiliser cette matrice pour projeter les données des autres vues v' sur l'espace engendré par ces vecteurs propres, les plus discriminants. Les auteurs obtiennent ainsi, pour une autre vue v', une nouvelle matrice de similarité $\mathbf{S}_{v'} = \mathbf{U}_v\,\mathbf{U}_v^\top\,\mathbf{S}'_v$. Cette opération est effectuée pour toutes les vues à chaque itération, et ce pour un nombre fixé d'itérations. Finalement, chaque matrice $\mathbf{U}_v$, de chaque itération, peut être utilisée pour obtenir une classification par l'algorithme des k-moyennes. Cette méthode est une extension de la méthode classification spectrale proposée par Ng *et al.* (2002).

Dans Tang *et al.* (2009), les auteurs considèrent qu'ils n'ont pas directement accès aux matrices de relations, et qu'il n'y a pas nécessairement de multiples types d'objets. Sont disponibles plusieurs matrices de similarités (ou d'adjacence, pondérées) pour un seul type d'objets, provenant de différentes sources (ou vues). Les auteurs proposent alors d'étendre les méthodes de factorisation de matrices à ce cas où de multiples matrices de similarité $\mathbf{S}_1, \ldots, \mathbf{S}_n$ sont disponibles. La fonction à optimiser est alors la suivante :

$$\sum_{v=1}^{N} \|\mathbf{S}_v - \mathbf{P}\,\mathbf{\Delta}_v\,\mathbf{P}^\top\|^2 + \text{terme de régularisation} \tag{4}$$

dans laquelle la matrice $\mathbf{P}$ de taille $n \times K$ est unique pour toutes les vues, et capture ainsi les facteurs communs aux différentes vues. Elle correspond aux affectations des objets aux classes. Les matrices $\mathbf{\Delta}_v$ de taille $K \times K$ quant à elles, capturent les différences entre les multiples vues. Le terme de régularisation permet lui d'éviter les instabilités numériques et le sur-apprentissage.

La fonction à maximiser définie dans l'Eq. (4) n'étant pas convexe, un algorithme d'optimisation locale est proposé par les auteurs, qui consiste à alternativement fixer la matrice $\mathbf{P}$ et à trouver les matrices $\mathbf{\Delta}_v$ maximisant Eq. (4), puis à fixer les $\mathbf{\Delta}_v$ et à trouver la matrice $\mathbf{P}$ maximisant Eq. (4), et ce jusqu'à stabilisation des solutions.

Dans le cadre des méthodes spectrales, Long *et al.* (2006a) privilégient la représentation sous forme de graphes multipartis des données. Le principe de cette approche est d'ajouter des nœuds cachés dans chaque couche de ce graphe, chacun de ces nœuds représentant une classe. L'objectif est alors de supprimer les arêtes initiales entre les nœuds du graphe, par des arêtes allant des nœuds représentant les objets aux nœuds cachés des classes, et également des arêtes entre les nœuds cachés des classes des différents types d'objets. Les arêtes allant des nœuds représentant les instances de X_i aux nœuds cachés peuvent être représentées par une matrice indicatrice ${}_i$ de taille $\eta_i \times K_i$. De plus, les arêtes entre les nœuds cachés des classes de X_i et des classes de X_j peuvent quant à elles être représentées par une matrice indicatrice $\mathbf{B}_{ij}$ de taille $K_i \times K_j$.

Long *et al.* (2006a) cherchent ensuite les matrices indicatrices minimisant la divergence de Bregman entre $\mathbf{R}$ et ${}_i\mathbf{B}_{ij}{}_j^\top$. Ce problème étant NP-complet, les auteurs proposent un algorithme d'optimisation locale. Finalement, si l'on ne garde que les nœuds cachés du graphe ainsi construit, on dispose d'un résumé du graphe multiparti de départ, d'où le nom de la méthode de *Relation Summary Network* (*RSN*).

I-3.5 Extensions de méthodes classiques au contexte multivue

Finalement, plusieurs extensions d'algorithmes de classification classiques ont depuis été proposées, afin de les adapter aux données multivues. Nous pouvons par exemple citer Bickel et Scheffer (2005), Drost *et al.* (2006) qui décrivent des extensions des algorithmes des k-moyennes et d'espérance-maximisation (EM). Dans Bickel et Scheffer (2005), les auteurs proposent une approche basée sur le calcul des valeurs de variables latentes dans une vue afin de les utiliser dans l'étape de maximisation pour l'autre vue, et *vice versa*.

Certaines approches quant à elles, commencent par extraire des caractéristiques communes pour le type d'objets à classifier, en se basant sur les corrélations entre les vecteurs représentant les autres types d'objets dans les différentes vues. Finalement, ces caractéristiques sont utilisées pour classifier les objets concernés à l'aide d'un algorithme de classification classique (Blaschko et Lampert 2008, Chaudhuri *et al.* 2009).

Dans de Carvalho *et al.* (2012), les auteurs proposent une méthode inspirée de l'algorithme de classification multivue *CARD* de Frigui *et al.* (2007) (voir Sec. I-3.1), et s'appuient donc

sur de multiples matrices de similarité. La différence avec $CARD$ est que leur approche permet d'apprendre une pondération des différentes vues et des classes w_{vk}. Cette méthode permet ainsi de pouvoir détecter qu'une vue donnée décrit mieux que les autres une classe en particulier, par exemple si le poids w_{vk} est élevé, cela signifie que la vue v décrit bien la classe k.

I-4 Méthodes de consensus de classifications

Il nous paraît finalement important de consacrer une partie de cet état de l'art aux méthodes de consensus de classifications visant, à partir d'un ensemble de classifications (typiquement des partitions) des mêmes objets, à trouver une classification unique. Ces méthodes ont été initialement développées dans le contexte monovue classique, afin d'obtenir plus de confiance dans les classes obtenues, en trouvant les classes les plus robustes, c'est à dire celles apparaissant fréquemment dans l'ensemble des classifications, produites à partir de sources différentes (différentes vues, différents algorithmes, différents paramètres, etc.). Intuitivement, la classification consensus doit partager le plus d'*information* possible avec les classifications à partir desquelles elle est construite. Il a été établi que le problème de la recherche de la classification consensus, aussi connu sous le nom de partition médiane, est NP-difficile (Wakabayashi 1998). Les méthodes présentées dans la suite de cette section sont donc des heuristiques pour traiter ce problème.

Cette famille de méthodes peut également être utilisée dans le cadre de l'apprentissage multivue de classifications. En effet, une classification est produite à partir de chacune des vues du problème, et une recherche de classification consensus peut ensuite être utilisée pour produire la classification finale.

Une autre application de cette famille de méthode évoquée par Strehl et Ghosh (2003), et très similaire au cadre multivue, concerne le cas où pour des questions de confidentialité, toutes les données ne sont pas détenues par une seule entité. Dans ce contexte, il est alors intéressant que chaque entité fournisse sa propre classification, et qu'un algorithme de consensus de classification soit ensuite utilisé pour trouver la classification finale. Une autre application peut être le calcul distribué, pour des masses de données trop importantes pour pouvoir être traitées de façon centralisée. Nous verrons dans le Chap. III, que notre contribution dans le cadre de l'apprentissage multivue de similarités, peut également être utile dans le cadre du calcul distribué, que ce soit pour des raisons de confidentialité, ou pour réduire la complexité en temps et mémoire du traitement des données.

Nous considérons ici que nous avons un ensemble de η objets $X = \{x_1, x_2, ..., x_\eta\}$ que nous souhaitons classifier en K classes. De plus, nous avons un ensemble de m classifications $\Pi = \{\pi_1, \pi_2, ..., \pi_m\}$, avec $\forall i \in [\![1...m]\!], \pi_i(x_j) \in [\![1...K]\!]$ représentant la classe affectée à l'objet j par la classification i. En effet, et comme dans le reste de ce manuscrit, nous supposons que les méthodes qui produisent les classifications connaissent le nombre de classes, et produisent donc K classes d'objets. L'objectif des méthodes présentées dans la suite de cette section est donc de trouver une partition finale à partir de Π, de façon à optimiser des critères différents.

On peut classer ces approches en trois catégories :

- celles basées sur la similarité entre les objets induite par Π que nous décrivons dans la Sec. I-4.1,
- celles basées sur le partitionnement d'un graphe construit à partir de Π que nous décrivons dans la Sec. I-4.2,
- et celles où l'on considère les classifications comme des caractéristiques des objets que nous décrivons dans la Sec. I-4.3.

I-4.1 Similarité induite par les classifications

Il est possible de construire, à partir de l'ensemble des classifications Π, une matrice de similarité entre les objets de X, en considérant l'idée simple suivante : si deux objets apparaissent souvent dans les mêmes classes, alors ils sont similaires. On construit ainsi la matrice de co-association, ou méta-matrice de similarité $\mathbf{B}$ de taille $\eta \times \eta$:

$$\mathbf{B} = [b_{ij}]_{1 \leqslant i,j \leqslant \eta} \text{ avec } b_{ij} = \frac{1}{m} \sum_{1 \leqslant l \leqslant m} \delta\left(\pi_l(x_i), \pi_l(x_j)\right) \tag{5}$$

où $\delta(a,b)$ vaut 1 si $a = b$ et 0 sinon, est la fonction indicatrice.

La matrice $\mathbf{B}$ est ainsi carrée et symétrique, et contient des valeurs comprises entre 0 et 1, c_{ij} valant 0 si x_i et x_j ne sont jamais classés ensemble, et 1 s'ils le sont toujours. Fred et Jain (2002) utilisent le terme de mécanisme de vote pour la construction de la matrice de co-association, car on peut considérer que chaque classification vote pour les paires d'objets qu'elle a classés ensemble.

Ainsi, à partir de cette méta-matrice de similarité, il est possible d'utiliser n'importe quel algorithme de classification (voir Sec. I-2) pour construire une classification consensus.

Par exemple, Fred et Jain (2002) produisent leurs multiples classifications à partir de différentes initialisations de l'algorithme des k-moyennes, puis construisent la matrice de co-association $\mathbf{B}$ qu'ils considèrent comme une *"accumulation de preuves"*, et cherchent ensuite les classes les plus stables en utilisant un algorithme de classification ascendante hiérarchique prenant $\mathbf{B}$ en entrée, et utilisant le saut minimum comme mesure de similarité inter-classe (voir Sec. I-2).

Fern et Brodley (2003) se situent dans un contexte différent, car ils cherchent à classifier des objets décrits par un très grand nombre de dimensions. Les auteurs utilisent alors des projections aléatoires de leurs données dans des espaces plus réduits, et construisent une classification pour chacune de ces projections. Ces classifications étant produites par un algorithme EM (Estimation-Maximisation, voir Sec. I-2), les $\pi_l(x_i)$ ne sont plus des entiers indiquant la classe de l'objet x_i par π_l, mais des distributions sur $1..K$. La similarité entre x_i et x_j est alors la moyenne des probabilités que ces deux objets soient dans la même classe. Leur matrice de co-association, est donc basée sur le même principe, et représente également une méta-matrice de similarité, que les auteurs utilisent pour produire leur classification finale à l'aide d'un algorithme de classification ascendante hiérarchique utilisant le saut maximum,

afin de ne pas regrouper ensemble des objets qui ne sont pas similaires, comme mesure de similarité inter-classe (voir Sec. I-2).

Comme pour les méthodes de classification et de co-classification, le domaine de la bio-informatique a proposé des méthodes de recherche de classification consensus. Par exemple, Monti *et al.* (2003) utilisent également une matrice de co-association, qu'ils appellent d'ailleurs *"matrice de consensus"*, et construisent leur classification finale à l'aide d'une classification ascendante hiérarchique utilisant le lien moyen comme mesure de similarité inter-classe (voir Sec. I-2).

I-4.2 Partitionnement de graphes induits par les classifications

Il existe différentes approches visant à trouver une classification consensus, basées sur le partitionnement de graphes créés à partir de l'ensemble des classifications Π (Strehl et Ghosh 2003, Fern et Brodley 2004). Au total, quatre graphes différents peuvent être définis à partir d'un ensemble de classifications (voir Fig. 8), et chacun de ces graphes peut être partitionné afin d'obtenir une classification consensus. Strehl et Ghosh (2003) définissent une fonction objectif à maximiser, qui représente la somme des informations mutuelles normalisées entre les classifications de Π et la classification consensus. Les auteurs proposent alors trois graphes différents, et Fern et Brodley (2004) proposent quant à eux la construction d'un quatrième graphe biparti.

CSPA

Pour la première méthode, on considère la matrice de co-association $\mathbf{B}$ décrite dans l'Eq. (5), et on crée le graphe basé sur les instances suivant $\mathcal{G}_1 = (\mathcal{D}_1,\mathcal{A}_1)$ avec :

- $\mathcal{D}_1$ l'ensemble des sommets du graphe représentant les η instances de X,
- et $\mathcal{A}_1$ l'ensemble des arêtes pondérées par les similarités de $\mathbf{B}$.

Il est ensuite possible d'utiliser un algorithme de partitionnement de graphe quelconque pour obtenir la classification finale. Dans Strehl et Ghosh (2003), les auteurs utilisent l'algorithme de partitionnement METIS (Karypis et Kumar 1999). Cette méthode est nommée CSPA pour *Cluster-Based Similarity Partitioning Algorithm* en anglais.

MCLA

Pour la deuxième méthode, on construit un graphe suivant basé sur les classes $\mathcal{G}_2 = (\mathcal{D}_2,\mathcal{A}_2)$ avec :

- $\mathcal{D}_2$ l'ensemble des sommets du graphe représentant les classes,
- et $\mathcal{A}_2$ l'ensemble des arêtes pondérées par les indices de Jaccard entre les classes. On rappelle comment calculer l'indice de Jaccard, qui mesure la similarité entre deux classes $\mathrm{c}_i, \mathrm{c}_j$: $\mathrm{Jaccard}(\mathrm{c}_i,\mathrm{c}_j) = \frac{\|\mathrm{c}_i \cap \mathrm{c}_j\|}{\|\mathrm{c}_i \cup \mathrm{c}_j\|}$.

Il est ensuite possible d'utiliser un algorithme de partitionnement de graphe quelconque pour obtenir une classification des classes. Les méta-classes (classes de classes) ainsi obtenues permettent de calculer la classification finale de la façon suivante : pour chaque objet x_i, un score d'appartenance à une méta-classe est calculée comme le nombre de classes de cette méta-classe où x_i apparaît. Finalement, cet objet est affecté à la méta-classe ayant obtenu le plus grand score. Cette méthode est nommée MCLA pour *Meta-CLustering Algorithm* en anglais.

HPGA

La troisième méthode est basée sur la construction de l'hypergraphe suivant $\mathcal{H}_3 = (\mathcal{D}_3,\mathcal{A}_3)$ avec :

- $\mathcal{D}_3$ l'ensemble des sommets du graphe représentant les η instances de X,
- et $\mathcal{A}_3$ l'ensemble des hyper-arêtes représentant les classes. Pour chaque classe de chaque classification π_i de l'ensemble Π, on crée l'hyper-arête reliant toutes les instances qu'elle contient.

Il est ensuite possible de créer la partition finale en utilisant un algorithme de partitionnement d'hyper-graphe quelconque. Dans Strehl et Ghosh (2003), les auteurs utilisent l'algorithme de partitionnement d'hyper-graphe HMETIS (Karypis *et al.* 1999), qui est une extension de METIS (Karypis et Kumar 1999) aux hyper-graphes. Cette méthode est nommée HGPA pour *HyperGraph Partitioning Algorithm* en anglais.

HBGF

La quatrième approche consiste à créer le graphe biparti suivant $\mathcal{G}_4 = ((\mathcal{D}_4^o,\mathcal{D}_4^c)),\mathcal{A}_4)$ avec :

- $\mathcal{D}_4^o$ les sommets du graphe représentant les η instances de X,
- $\mathcal{D}_4^c$ les sommets du graphe représentant les classes,
- et $\mathcal{A}_4$ l'ensemble des arêtes représentant l'appartenance des instances aux classes. Comme il s'agit d'un graphe biparti, il n'y a pas d'arête entre les sommets de $\mathcal{D}_4^o$, et pas d'arête entre ceux de $\mathcal{D}_4^c$.

Finalement, et comme dans les approches précédentes, il est possible d'utiliser un algorithme de partitionnement de graphe biparti quelconque pour obtenir la classification finale. Dans Fern et Brodley (2004), les auteurs testent leur méthode avec deux algorithmes de partitionnement différent : METIS (Karypis et Kumar 1999) et SPEC (Ng *et al.* 2002), un algorithme spectral. Cette méthode est nommée HBGF pour *Hybrid Biparti Graph Formulation* en anglais.

Exemple illustratif

Afin d'illustrer ces différentes méthodes, considérons l'exemple suivant : $X = \{x_1, ..., x_6\}$, $\Pi = \{\pi_1, ..., \pi_3\}$ avec les classifications décrites par la Tab. 1. La Fig. 8 présente les 4 types de graphes sur lesquels sont basées les méthode précédentes.

	π_1	π_2	π_3
x_1	1	1	1
x_2	2	1	2
x_3	3	2	3
x_4	1	3	1
x_5	2	3	3
x_6	3	2	3

TABLE 1 – Exemple illustratif d'un ensemble de classifications d'objets ($\eta = 6$, $K = 3$ et $m = 3$)

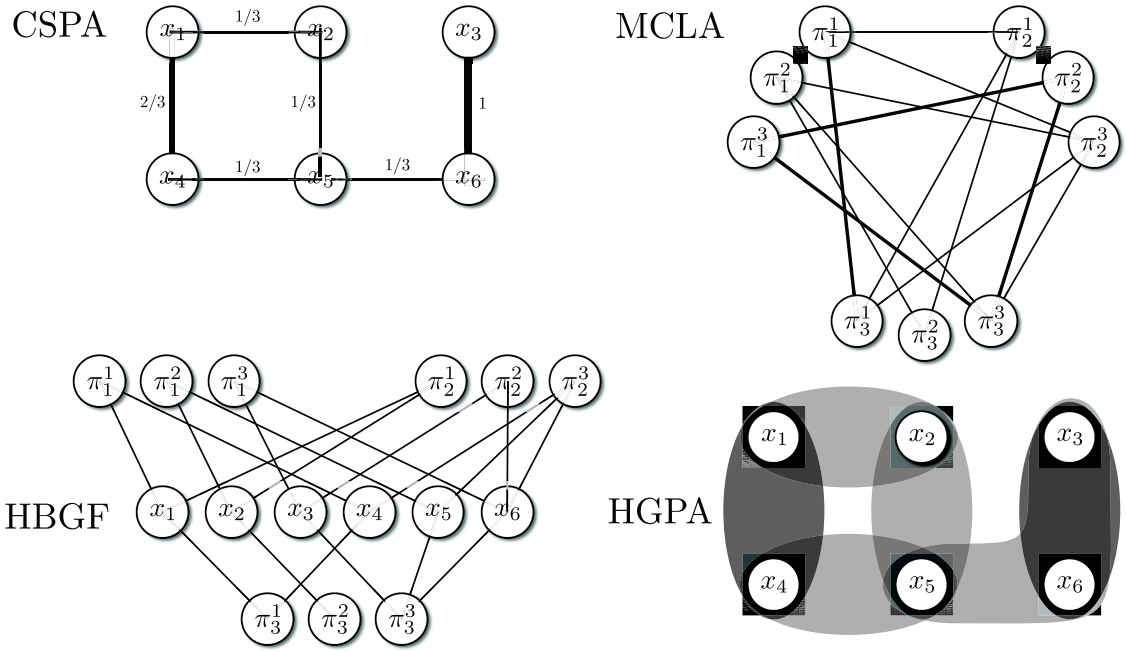


FIGURE 8 – Illustration de graphes pour l'exemple présenté dans la Tab. 1.

Comparaisons de ces quatre approches

L'algorithme CSPA est le seul à avoir une complexité temporelle quadratique en $\mathcal{O}(\eta^2\, K\, m)$ et bien qu'il soit le plus simple et le plus intuitif, il n'est en pratique pas utilisable pour des jeux de données de grande taille. Les algorithmes les plus rapides sont HPGA et HBGF qui ont une complexité en $\mathcal{O}(\eta\, K\, m)$, suivi de près par MCLA qui a une complexité temporelle en $\mathcal{O}(\eta\, K^2\, m^2)$, car dans la plupart des applications $K \ll \eta$.

Pour conclure, la méthode HBGF proposée par Fern et Brodley (2004) semble être la plus intéressante car la représentation sous forme de graphe biparti n'implique aucune perte d'information, dans le sens où il est possible de retrouver toutes les classifications à partir de celui-ci. De plus, les auteurs ont obtenu de meilleurs résultats avec HBGF qu'avec les méthodes proposées par Strehl et Ghosh (2003). Rappelons également que cette méthode est l'une des plus rapides.

I-4.3 Caractéristiques induites par les classifications

Une autre approche consiste à considérer les classifications comme autant de nouvelles caractéristiques des objets (Topchy *et al.* 2003, 2005). Chaque partition π_i est alors vue comme une nouvelle caractéristique catégorielle des données X. On obtient ainsi m nouvelles caractéristiques, et généralement, afin de pouvoir utiliser un algorithme de classification de données catégorielles, les auteurs n'utilisent pas les caractéristiques originales.

Les auteurs proposent de travailler dans cet espace de caractéristiques induites par les classifications de Π, mais reformulent le problème de recherche de la classification consensus en un problème d'estimation du maximum de vraisemblance, sur un modèle de mélange basé sur l'hypothèse que les classes peuvent être modélisées comme les variables tirées à partir des

densités d'un modèle de mélange multivarié. Le problème est ensuite traité par un algorithme EM (Estimation-Maximisation, voir Sec. I-2). Cette méthode est appelée QMI pour *Quadratic Mutual Information* en anglais.

Topchy *et al.* (2003, 2005) génèrent de multiples classifications de deux façons différentes :

- grâce à de multiples instances de l'algorithme des k-moyennes, utilisées dans des espaces projetés aléatoirement,
- grâce à des divisions des données à l'aide d'hyperplans aléatoires.

Ces classifications peuvent être générées de façon très rapide, et sont relativement simples, mais les auteurs prouvent que même à partir de classifications « faibles », leur méthode obtient de bons résultats, grâce à la combinaison d'un grand nombre d'entre elles. Une classification est dite « faible » si elle ne classe les objets de X que de façon légèrement meilleure qu'une classification aléatoire. La justification apportée par les auteurs est que la synergie de ces multiples classifications compense alors leur « faiblesse ».

I-4.4 Autres méthodes

Finalement, il existe d'autres méthodes pour la recherche de classification consensus, qui utilisent des approches différentes (Dimitriadou *et al.* 2001, Dudoit et Fridlyand 2003, Li et Ding 2008).

Dimitriadou *et al.* (2001), Dudoit et Fridlyand (2003) utilisent un mécanisme de vote, et contrairement aux approches précédemment décrites, cherchent à résoudre le problème de la correspondance des classes. A l'instar de la méthode CSPA, ces deux approches sont basées sur la construction d'une méta-matrice de similarité construite à partir des multiples classifications (obtenues pour différents algorithmes de classification (Dimitriadou *et al.* 2001) ou pour différents échantillons des données (Dudoit et Fridlyand 2003)). Pour chaque objet, chacune des partitions vote alors pour une classe, et c'est finalement la classe majoritaire qui est élue comme classe « gagnante ». De plus, il est alors possible d'évaluer la confiance de cette classification en observant, pour chacun des objets, la proportion de classifications de l'ensemble Π ayant voté pour la classe « gagnante ».

En se basant toujours sur cette notion de méta-matrice de similarité, Li et Ding (2008) proposent de chercher une factorisation non-négative de cette matrice notée $\widetilde{\mathbf{B}}$:

$$\min_{\mathbf{E}} \|\widetilde{\mathbf{B}} - \mathbf{E}\,\mathbf{E}^{\top}\|^2 \tag{6}$$

où $\mathbf{E}$ est la matrice d'affectation de taille $\eta \times K$ qui associe une classe à chaque objet, avec donc la contrainte que chaque ligne ne doit contenir qu'un seul 1. De plus, les auteurs cherchent à pondérer les différentes classifications de Π, ce qui modifie légèrement la construction de la matrice de co-association $\widetilde{\mathbf{B}}$ par rapport à l'Eq. (5) :

$$\widetilde{\mathbf{B}} = \left[\tilde{b_{ij}}\right]_{1\leqslant i,j\leqslant \eta} \text{ avec } \tilde{b_{ij}} = \frac{1}{K} \sum_{1\leqslant l\leqslant K} \alpha_l\, \delta\left(\pi_l(x_i), \pi_l(x_j)\right)$$

où α_l est le poids associé à la classification π_l. Le problème est ensuite résolu en alternant les deux étapes suivantes :

- les poids des classifications α_l étant fixés, résolution de (6) pour trouver $\mathbf{E}$,
- la matrice $\mathbf{E}$ étant fixée, résolution de (6) pour trouver les poids α_l.

Cette méthode est à rapprocher des approches de classification spectrale décrites dans la Sec. I-2.3.d, et peut en conséquence être potentiellement améliorée par les développements de ces approches.

I-5 Évaluation des classifications produites

Il existe de nombreuses mesures d'évaluation de classification, que l'on peut regrouper en deux familles, selon que l'on cherche à effectuer :

- Une **évaluation interne** : on évalue la classification produite en se basant uniquement sur les données. Il est par exemple possible de mesurer la cohérence des classes, en calculant les similarités entre les objets appartenant à une même classe, et celles entre les objets appartenant à des classes différentes.
- Une **évaluation externe** : on évalue la classification produite en s'appuyant sur des informations (comme les classes attendues des objets classifiés) n'ayant pas été utilisées par l'algorithme de classification.

Dans le cas de l'évaluation interne, on cherche à évaluer l'étape (3) du processus de classification présenté dans l'Introduction, alors que dans le cas de l'évaluation externe, on évalue l'ensemble du processus.

Dans le cadre de cette thèse, nous nous restreignons à l'étape (2) qui consiste à calculer les mesures de similarité entre les objets à classifier. De plus, et afin de pouvoir comparer nos approches avec des approches existantes, nous choisirons toujours le même algorithme de classification pour effectuer l'étape (3). En faisant cela, nous faisons alors l'hypothèse que les performances des différentes mesures de similarités seront comparables, indépendamment de l'algorithme choisi pour l'étape (3). Dans la suite de ce manuscrit, nous utiliserons alors uniquement une évaluation externe, en partant de jeux de données déjà classifiés, dont les classes attendues pourront être utilisées pour la phase d'évaluation.

Ainsi, dans la suite de cette section, nous allons décrire quatre méthodes classiques d'évaluation de classification :

- La matrice de confusion dans la Sec. I-5.1,
- La précision micro-moyennée (Pr) dans la Sec. I-5.2,
- L'information mutuelle normalisée (NMI) dans la Sec. I-5.3,
- L'entropie (H) dans la Sec. I-5.4,
- l'indice de Rand ajusté (ARI) dans la Sec. I-5.5.

I-5.1 La matrice de confusion

La plupart des mesures d'évaluation de classification sont basées sur la matrice de confusion, qui n'est pas elle-même une mesure, mais qui permet de rapidement visualiser les résultats d'une classification. D'une manière générale, soit K le nombre de classes notées $\mathrm{l}_1 \dots \mathrm{l}_K$, alors

la matrice de confusion, notée $\mathbf{C}$ sera, pour une classification des objets en L classes, de taille $K \times L$. Les lignes de $\mathbf{C}$ représentent les classes réelles et les colonnes de $\mathbf{C}$ représentent les classes prédites. Ainsi, le terme c_{ij} de la matrice de confusion est égal au nombre d'objets appartenant à la classe l_i et ayant été classifiés comme appartenant à la classe l_j. De plus, $c_{i:}$ est utilisé pour représenter la somme des termes de la ligne i de la matrice de confusion, soit le nombre d'objets de classe l_i. De façon duale, $c_{:j}$ représente le nombre d'objets ayant été classifiés comme appartenant à la classe l_j. On rappelle également que N est le nombre total d'objets, égal à la somme de tous les termes de $\mathbf{C}$.

Dans un contexte d'apprentissage non supervisé, les classes que l'on obtient ne sont pas étiquetées, c'est à dire que lors de la comparaison avec les classes attendues des objets, on ne connaît pas le couplage entre classes prédites et réelles. En d'autres termes, rien ne nous garantit que la classe prédite i corresponde à la classe réelle l_i. Ainsi pour certaines mesures d'évaluation, on cherche à associer à chaque classe prédite, la « meilleure » classe réelle. Ce problème est connu sous le nom de problème d'affectation en recherche opérationnelle, et il peut être résolu en temps polynomial par l'algorithme hongrois (Kuhn 1955). Plus concrètement, cela revient à maximiser les termes diagonaux de la matrice de confusion $\mathbf{C}$.

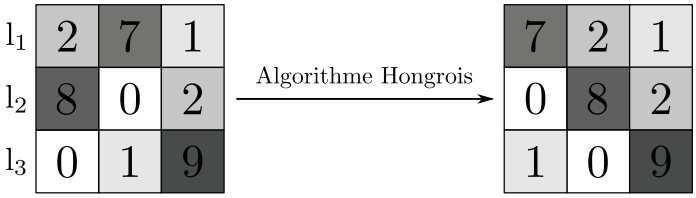


FIGURE 9 – Exemple de matrice de confusion $\mathbf{C}$ pour $N = 30$ et $K = L = 3$, avant et après optimisation par l'algorithme Hongrois.

La Fig. 9 représente un exemple de matrice de confusion pour $N = 10$ et $K = L = 3$, avant optimisation par l'algorithme Hongrois sur la gauche de la figure, puis après optimisation sur la partie droite, où les deux premières classes prédites ont été inversées, pour maximiser les poids se trouvant sur la diagonale de $\mathbf{C}$.

Un des intérêts de la matrice de confusion réside également dans le fait qu'elle présente de façon synthétique les erreurs de la classification produite[5]. On verra par exemple immédiatement si l'algorithme a tendance à créer des classes déséquilibrées, ou s'il a tendance à bien savoir séparer certaines classes mais en confond d'autres.

I-5.2 La précision micro-moyennée

La précision micro-moyennée correspond au pourcentage de documents correctement classifiés, et elle est généralement utilisée pour évaluer tous types de tâches d'apprentissage, aussi bien dans un contexte supervisé, qu'en fouille de données, et également en classification (Dhillon *et al.* 2003, Long *et al.* 2005). Sa formule est la suivante :

5. A condition que le nombre de classes soit relativement faible, ce qui est le cas dans la plupart des applications.

$$\Pr(\mathbf{C}) = \frac{1}{N} \sum_{i=1}^{K} c_{ii}$$

La précision micro-moyennée est une valeur comprise entre 0 et 1, avec une valeur de 1 si la matrice de confusion est diagonale et par conséquent la classification parfaite.

Cette mesure est beaucoup utilisée car elle est très intuitive, mais elle a cependant un défaut majeur : elle peut potentiellement donner une fausse impression de qualité de la classification. Concrètement, bien que sa valeur minimale théorique soit 0, à partir du moment où l'algorithme Hongrois est utilisé pour optimiser la matrice de confusion avant le calcul de Pr, elle ne pourra pas être inférieure à $\frac{\max_{i \in 1..K} |X_i|}{N}$. Ce défaut sera d'autant plus gênant dans le cas d'un problème où les classes ne sont pas équilibrées. Si par exemple la plus grande classe d'un problème contient 80% des objets, alors un algorithme qui construirait une classe contenant tous les objets du jeu de données, obtiendra une précision micro-moyennée de 80%.

I-5.3 L'information mutuelle normalisée

L'information mutuelle normalisée (Banerjee et Ghosh 2002, Strehl et Ghosh 2003), est un peu moins intuitive que la précision micro-moyennée, mais n'a pas ses défauts. Elle se calcule à partir de la matrice de confusion de la façon suivante :

$$\mathrm{NMI}(\mathbf{C}) = \frac{\sum_{i=1}^{K} \sum_{j=1}^{K} c_{ij} \log\left(\frac{c_{ij}}{c_{i:}\, c_{:j}}\right)}{\sqrt{\left(\sum_{i=1}^{K} c_{i:} \log(\frac{c_{i:}}{N})\right)\left(\sum_{j=1}^{K} c_{:j} \log(\frac{c_{:j}}{N})\right)}}$$

L'information mutuelle normalisée est une valeur comprise entre 0 et 1 également, mais qui elle ne nécessite pas l'optimisation de la matrice de confusion par l'algorithme Hongrois. Et contrairement à la précision micro-moyennée, l'information mutuelle normalisée d'une classification aléatoire vaut 0. La valeur 1 est également atteinte pour la classification parfaite, quand $\mathbf{C}$ est diagonale.

L'information mutuelle normalisée correspond à l'information mutuelle entre la distribution des classes prédites et des classes réelles, normalisée par le produit des entropies de ces distributions. Pour deux distributions X et Y, on la trouve le plus souvent notée comme suit :

$$\mathrm{NMI}(X,Y) = \frac{\mathrm{I}(X,Y)}{\sqrt{\mathrm{H}(X)\,\mathrm{H}(Y)}}$$

avec I désignant l'information mutuelle et H désignant l'entropie.

I-5.4 L'entropie

L'entropie est une mesure liée à l'incertitude, que l'on associe dans de nombreux contextes, à une quantification du désordre. Elle peut ainsi être utilisée pour mesurer le désordre dans les classes prédites, entre les objets des classes réelles (Bickel et Scheffer 2005, Drost *et al.* 2006). En pratique, on calcule l'entropie des classes réelles au sein des classes prédites, à l'aide de la formule suivante :

$$\mathrm{H}(\mathbf{C}) = \sum_{j=1}^{L} \frac{c_{:j}}{N} \left(-\sum_{i=1}^{K} \frac{c_{ij}}{c_{:j}} \log(\frac{c_{ij}}{c_{:j}}) \right)$$

L'entropie pour une matrice de confusion est une valeur entre 0 et $\log(K)$, et ne nécessite pas non plus l'optimisation de la matrice de confusion par l'algorithme Hongrois. Contrairement aux deux mesures précédentes, on cherche à la minimiser car 0 représente la classification parfaite, pour laquelle les classes prédites sont pures.

De façon intuitive, l'entropie est alors le nombre moyen de bits nécessaire pour encoder la classe réelle d'un objet à partir de sa classe prédite.

I-5.5 L'indice de Rand ajusté

Afin de comparer deux partitions entre elles, l'indice de Rand (Hubert et Arabie 1985) considère les objets par paire, et répartit ses paires en quatre ensembles :

- l'ensemble des paires d'objets qui sont dans la même classe dans les deux partitions, dont le cardinal est noté N_{11} ;
- l'ensemble des paires d'objets qui sont dans des classes différentes dans les deux partitions, dont le cardinal est noté N_{00} ;
- l'ensemble des paires d'objets qui sont dans la même classe dans la première partition, et dans des classes différentes dans la seconde, dont le cardinal est noté N_{10} ;
- l'ensemble des paires d'objets qui sont dans des classes différentes dans la première partition, et dans la même classe dans la seconde, dont le cardinal est noté N_{01}.

Les deux premières valeurs N_{11}, N_{00} permettent intuitivement de quantifier l'accord entre les deux partitions, tandis que les deux suivantes N_{10}, N_{01} quantifient au contraire le désaccord entre ces deux partitions.

A partir de ces quantités, l'indice de Rand est défini comme suit :

$$\mathrm{Rand} = \frac{N_{11} + N_{00}}{N_{11} + N_{00} + N_{10} + N_{01}}$$

Cependant, bien que cette mesure soit comprise entre 0 et 1, Hubert et Arabie (1985) remarquent que la valeur 0 n'est atteinte que dans le cas extrême où l'une des partitions est composée d'une seule classe contenant tous les objets, et la seconde partition est composée d'autant de classes que d'objets. De plus, ils notent que lorsque l'on compare deux partitions aléatoires des objets, les valeurs prises par l'indice de Rand sont très variables, et sont rarement proches de 0, ce qui serait souhaitable.

Afin de corriger ce problème de l'indice de Rand, Hubert et Arabie (1985) proposent alors de le corriger pour mieux prendre en compte la « chance » dans la classification, en faisant l'hypothèse d'une distribution hypergéométrique des données. Le nouvel index corrigé est appelé indice de Rand ajusté, ou ARI de l'anglais *Adjusted Rand Index* et se calcule à partir

de la matrice de confusion $\mathbf{C}$ de la façon suivante :

$$\mathrm{ARI}(\mathbf{C}) = \frac{\sum_{i=1}^{K}\sum_{j=1}^{L}\binom{c_{ij}}{2} - \left(\sum_{i=1}^{K}\binom{c_{i:}}{2}\sum_{j=1}^{L}\binom{c_{:j}}{2}\right)/\binom{N}{2}}{\frac{1}{2}\left(\sum_{i=1}^{K}\binom{c_{i:}}{2} + \sum_{j=1}^{L}\binom{c_{:j}}{2}\right) - \left(\sum_{i=1}^{K}\binom{c_{i:}}{2}\sum_{j=1}^{L}\binom{c_{:j}}{2}\right)/\binom{N}{2}}$$

Ainsi, l'indice de Rand ajusté est une valeur comprise entre 0 et 1 et qui ne nécessite pas l'optimisation de la matrice de confusion par l'algorithme Hongrois. Sa valeur est proche de 0 lorsque l'on compare une classification aléatoire avec la classification réelle des objets, et vaut 1 si les deux classifications sont les mêmes.

Chapitre II

Etude de l'algorithme χ-Sim

Sommaire du chapitre

C'est dans le cadre de la co-classification automatique de deux types d'objets que se situe la première contribution de cette thèse. Nous avons présenté dans le chapitre précédent les avantages des méthodes de co-classification, même dans le cas où l'utilisateur n'est intéressé par la classification que d'un seul type d'objets, et en particulier le fait que cette famille d'approches permettent de surmonter le fléau de la dimensionnalité.

Dans ce chapitre, nous commençons par présenter l'algorithme de calcul de co-similarité χ-Sim proposé par Bisson et Hussain (2008). Nous nous attacherons en particulier à décrire son fonctionnement dans le cadre d'une tâche de classification de documents décrits par les mots qui les composent, tout en sachant que son application ne se limite en aucun cas à cette tâche. Cet algorithme se base sur la dualité de la relation entre les documents et les mots, chaque type d'objets pouvant être décrit par l'autre. Bien que cet algorithme ait exhibé des performances intéressantes, il souffre de quelques défauts pratiques – comme le fait que la similarité d'un élément à lui même ne soit pas nécessairement maximale – et manque de justifications théoriques.

C'est pourquoi, nous présentons dans ce chapitre une amélioration de l'algorithme χ-Sim, qui concerne son schéma de normalisation, ainsi que l'introduction d'une pseudo-norme, généralisant la mesure. Par ce biais, nous prouvons que les matrices de similarités calculées par χ-Sim sont semi-définies positives, obtenant ainsi une plus solide base pour la justification et l'interprétation de l'algorithme. De plus, nous présentons des pistes en vue d'obtenir d'autres améliorations, comme l'amortis-

sement des similarités d'ordre supérieur, ainsi qu'un problème d'optimisation associé, afin de tenter de fournir une meilleure justification théorique à l'algorithme.

II-1 Présentation de l'algorithme χ-Sim original

Dans ce chapitre, nous considérons que les données sont décrites par une seule vue, et donc une seule matrice de relation $\mathbf{R}$ (nous ne notons plus la vue en indice ou en exposant pour alléger les notations), décrivant la relation entre deux types d'objets X_1 et X_2. Pour plus de simplicité, et pour rendre les explications plus illustratives, nous supposerons que X_1 représente des documents, et X_2 des mots, et donc $\mathbf{R}$ est une matrice de co-occurrences. Bien sûr, l'algorithme χ-Sim est applicable à d'autres types de données, comme par exemple les matrices d'expressions géniques. Nous nous intéresserons donc au calcul de deux matrices de similarités :

- $\mathbf{S}_1$, de taille $\eta_1 \times \eta_1$ qui contient les similarités entre toutes les paires d'instances de X_1,
- $\mathbf{S}_2$, de taille $\eta_2 \times \eta_2$ qui contient les similarités entre toutes les paires d'instances de X_2.

Rappelons que ces matrices de similarité sont carrées, symétriques et que leurs valeurs sont dans l'intervalle [0,1].

Dans la suite de cette section, nous présenterons l'algorithme de calcul de co-similarité initialement proposé par Bisson et Hussain (2008), puis dans les sections suivantes de ce chapitre, nous présenterons des améliorations possibles de cette méthode.

II-1.1 Intuition et idée de départ

L'idée fondamentale sur laquelle repose cette approche est que pour être similaires, deux documents n'ont pas nécessairement besoin d'avoir des mots en communs, comme c'est le cas pour toutes les mesures classiques vues dans la Sec. I-2.1.b, mais seulement des mots similaires. Ainsi, lorsque l'on cherche à évaluer la similarité entre deux documents, l'idée est donc de ne pas seulement prendre en compte les mots communs, mais également toutes les paires de mots similaires qui y apparaissent. Respectivement, deux mots sont similaires s'ils apparaissent dans des documents similaires. On remarque immédiatement la dualité entre les documents et les mots, et la symétrie dans leur relation. Le problème qui se dégage immédiatement de cette idée de départ est que les similarités entre documents dépendent des similarités entre mots, et *vice versa*.

χ-Sim est donc une approche de co-similarité qui part de l'idée de simultanément calculer les deux matrices de similarité $\mathbf{S}_1$ (documents) et $\mathbf{S}_2$ (mots), et surtout, de calculer l'une par rapport à l'autre. Des idées similaires ont été utilisées dans le cadre de l'apprentissage supervisé (Liu *et al.* 2004), ou pour une tâche de recherche d'images (Wang *et al.* 2004).

Dans le contexte de la classification automatique de documents textuels, la mesure de similarité entre deux documents x_i et x_j généralement privilégiée est la mesure du Cosinus, présentée dans la Sec. I-2.1.b, dont nous rappelons ici sa formule :

$$\mathrm{Cos}(x_i,x_j) = \frac{x_i \cdot x_j}{\|x_i\| \, \|x_j\|} \tag{7}$$

Le principal défaut de cette mesure de similarité est qu'elle s'appuie sur le produit scalaire entre les deux vecteurs caractérisant ces documents, et donc uniquement sur les mots en commun entre ces documents. En effet, cette méthode ne prend pas en compte le fait que certains mots soient similaires, voire des synonymes. Nous pouvons illustrer ce problème à l'aide de l'exemple de deux extraits du livre *« Le vin j'y connais rien »* de Paumard, Millet et Gabs, issus du chapitre expliquant comment sentir un vin, mais utilisant des vocabulaires légèrement différents :

Apprendre à sentir un vin est un processus long et délicat, car cela nécessite d'engranger de multiples fragrances.

Les expressions aromatiques variétales sont typiques de chaque cépage, et seuls les amateurs rodés à l'art subtil de l'olfaction parviennent à les reconnaître.

On remarque que ces deux textes n'ont aucun terme en commun : *vin* est pourtant proche de *cépage*, *sentir* similaire à *olfaction*, et *fragrances* un synonyme d'*expressions aromatiques*. Même les adjectifs *délicat* et *subtil* sont proches d'un point de vue sémantique. Pourtant, en utilisant la mesure de similarité du Cosinus, ou tout autre mesure classique présentée dans la Sec. I-2.1, on trouvera que ces deux textes ont une similarité nulle (ou une distance maximale).

Partant de ce défaut des mesures classiques, les méthodes de co-classification (voir Sec. I-2.3) cherchent à classifier les mots en même temps que les documents, afin de palier ce problème. Si par exemple les termes *vin* et *cépage* étaient regroupés dans une même classe, alors il serait plus aisé de voir que les deux textes sont en effet similaires. L'approche de la mesure de co-similarité χ-Sim adopte cette vision, mais elle diffère en cela qu'elle ne classe pas directement les documents et les mots, mais qu'elle calcule leurs similarités.

Notons que le fait de calculer ces deux matrices de similarités $\mathbf{S}_1$ et $\mathbf{S}_2$ permet de facilement obtenir une co-classification des instances de X_1 et de X_2. En effet, il suffit pour cela d'utiliser un algorithme de classification classique (voir Sec. I-2.2) sur chacune de ces deux matrices de similarités, puis d'utiliser un algorithme d'affectation comme l'algorithme Hongrois (voir Sec. I-5.1) pour associer à chaque classe de X_1 la classe de X_2 avec laquelle elle sera liée. Se concentrer sur le calcul des mesures de similarité offre au moins deux avantages :

- laisser l'utilisateur libre de choisir l'algorithme de classification qu'il préfère,
- permettre l'utilisation de ces similarités dans d'autres contextes, pour d'autres applications que la classification.

II-1.2 Calcul des co-similarités

Dans cette partie, nous allons voir comment χ-Sim calcule la similarité entre toutes les paires de documents (lignes) et mots (colonnes) d'une matrice de relation $\mathbf{R}$. Afin d'illustrer plus clairement les explications, nous nous attarderons sur le calcul de la similarité entre les documents x_i et x_j, respectivement représentés par les lignes (ou vecteurs) $\mathbf{r}_{i:}$ et $\mathbf{r}_{j:}$, et au calcul de la matrice $\mathbf{S}_1$.

Nous introduisons ici une fonction générique de similarité élémentaire $\mathrm{F_s}(\cdot, \cdot)$, prenant en argument deux éléments d'une matrice de relation, qui calcule un score de similarité entre ces

deux valeurs (pas nécessairement entre 0 et 1). Cette fonction dépend du type – numérique, catégoriel, etc. – de ces arguments. Dans le cas numérique, elle dépend alors également du modèle de pondération utilisé (voir Sec. I-1.1.b), et il s'agit généralement d'une opération algébrique comme le produit. Dans le cas d'arguments catégoriels, on utilise généralement une matrice de similarité entre les différentes valeurs possibles de ces arguments.

De façon classique, la similarité (ou la distance) entre deux documents $\mathbf{r}_{i:}$ et $\mathbf{r}_{j:}$ est définie comme une fonction, dénotée ici $\mathrm{Sim}(\mathbf{r}_{i:},\mathbf{r}_{j:})$, impliquant uniquement la somme des similarités élémentaires entre les mots qu'ils contiennent :

$$\begin{aligned}\mathrm{Sim}(\mathbf{r}_{i:},\mathbf{r}_{j:}) &= \mathrm{F_s}(r_{i1},r_{j1}) + \cdots + \mathrm{F_s}(r_{i\eta_2},r_{j\eta_2}) \\ &= \sum_{j=1}^{\eta_2} \mathrm{F_s}(r_{ik},r_{jk})\end{aligned}$$

Comme nous l'avons vu avec l'exemple des deux textes sur le vin, cela peut ne pas être judicieux car si $\mathbf{r}_{i:}$ contient un mot similaire à un mot de $\mathbf{r}_{j:}$, alors il est important de capturer cette contribution afin d'augmenter la similarité entre ces deux documents. Supposons donc que nous disposions d'une mesure de la similarité entre toutes les paires de mots du vocabulaire : la matrice $\mathbf{S}_2$. Le terme $s_{kl}^{(2)}$ est ainsi la similarité entre les termes k et l. Maintenant, il est possible d'utiliser ces similarités entre termes pour pondérer les contributions de chaque paires de mots apparaissant dans les documents :

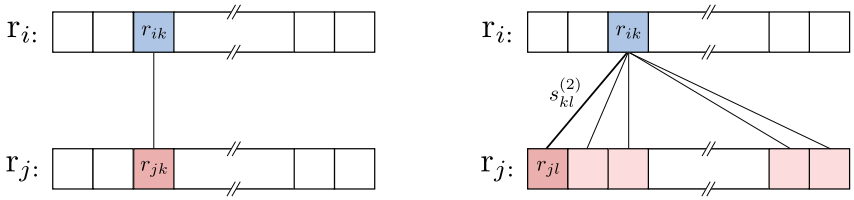
$$\mathrm{Sim}(\mathbf{r}_{i:},\mathbf{r}_{j:}) = \sum_{k=1}^{\eta_2}\sum_{l=1}^{\eta_2} \mathrm{F_s}(r_{ik},r_{jl}) \times s_{kl}^{(2)} \tag{8}$$

Reprenons les deux textes sur le vin, si nous savons, grâce à la matrice $\mathbf{S}_2$ que *vin* et similaire à *cépage* et que *sentir* et similaire à *olfaction*, alors la similarité entre ces deux textes ne sera plus nulle. Par conséquent, la similarité entre deux documents ne prend plus seulement en compte les mots qu'ils ont en commun, mais bien toutes les paires de mots, même s'ils ne sont pas directement partagés par ces documents. Remarquons au passage qu'en prenant $\mathbf{S}_2 = \mathbf{I}$, on retrouve la forme générale d'une mesure de similarité.

La Fig. 10 fournit une illustration de cette approche. En effet, on voit sur le schéma (a) qui représente une mesure classique de similarité, que seuls sont comparés les éléments ayant le même indice de colonne, alors que sur le schéma (b), tous les éléments sont comparés entre eux, pondérés par les similarités entre les colonnes (mots).

En supposant maintenant que la fonction de similarité élémentaire $\mathrm{F_s}$ est simplement le produit de ces deux arguments (comme pour la mesure du Cosinus), il est possible de réécrire l'Eq. (8) sous la forme d'un produit vecteurs matrice :

$$\mathrm{Sim}(\mathbf{r}_{i:},\mathbf{r}_{j:}) = \mathbf{r}_{i:} \times \mathbf{S}_2 \times \mathbf{r}_{j:}^{\top}$$



(a) Mesure de similarité classique. (b) Approche χ-Sim

Figure 10 – Comparaison des mesures classiques de similarité avec l'approche de χ-Sim.

Il est important de noter que le fait d'utiliser le produit comme fonction de similarité a un sens pour les différents schémas de pondération présentés Sec. I-1.1.b. Pour le modèle binaire, le produit correspond à un ET logique, et pour le nombre d'occurrences et le *TF-IDF*, cela donne plus de poids si les mots apparaissent souvent. De plus, le fait d'utiliser le produit permet de calculer toutes les similarités en une seule opération, réduisant ainsi la complexité de l'approche d'un facteur η_2 en réalisant le produit matriciel suivant :

$$\mathbf{R} \times \mathbf{S}_2 \times \mathbf{R}^\top$$

Finalement, il est nécessaire de normaliser cette mesure de similarité afin qu'elle appartienne à l'intervalle [0,1], on introduit alors la fonction de normalisation $\mathcal{N}(\mathbf{r}_{i:},\mathbf{r}_{j:})$:

$$\mathrm{Sim}(\mathbf{r}_{i:},\mathbf{r}_{j:}) = \frac{\mathbf{r}_{i:} \times \mathbf{S}_2 \times \mathbf{r}_{j:}^\top}{\mathcal{N}(\mathbf{r}_{i:},\mathbf{r}_{j:})} \tag{9}$$

L'équation (9) est la forme générique de différentes mesures de similarité que nous avons présentées dans la Sec. I-2.1. En effet, selon la manière dont sont définis $\mathbf{S}_2$ et $\mathcal{N}$, on retrouve les mesures suivantes :

- l'indice de Jaccard pour $\mathbf{S}_2 = \mathbf{I}$ et $\mathcal{N}(\mathbf{r}_{i:},\mathbf{r}_{j:}) = \|\mathbf{r}_{j:}\|^2 \|\mathbf{r}_{j:}\|^2 - \mathbf{r}_{i:} \cdot \mathbf{r}_{j:}$.
- le coefficient de Dice pour $\mathbf{S}_2 = \mathbf{I}$ et $\mathcal{N}(\mathbf{r}_{i:},\mathbf{r}_{j:}) = \|\mathbf{r}_{j:}\|^2 \|\mathbf{r}_{j:}\|^2$.
- la similarité du Cosinus pour $\mathbf{S}_2 = \mathbf{I}$ et $\mathcal{N}(\mathbf{r}_{i:},\mathbf{r}_{j:}) = \sqrt{\|\mathbf{r}_{i:}\|_2 \|\mathbf{r}_{j:}\|_2}$.
- la similarité du Cosinus généralisé (Qamar et Gaussier 2009) pour $\mathbf{S}_2$ semi-définie positive (abrégé en SDP dans la suite de ce manuscrit), définissant la norme $\|\mathbf{r}_{i:}\|_{\mathbf{S}_2}$ et $\mathcal{N}(\mathbf{r}_{i:},\mathbf{r}_{j:}) = \sqrt{\|\mathbf{r}_{i:}\|_{\mathbf{S}_2} \|\mathbf{r}_{j:}\|_{\mathbf{S}_2}}$ (plus de détails dans la Sec. II-2.1).

Dans l'algorithme χ-Sim initial (Bisson et Hussain 2008), le schéma de normalisation choisi représente le produit des longueurs des deux documents, ce qui permet (comme les autres normalisations présentées) de pouvoir comparer des documents de tailles différentes : $\mathcal{N}(\mathbf{r}_{i:},\mathbf{r}_{j:}) = |\mathbf{r}_{i:}| \times |\mathbf{r}_{j:}|$. Pour des questions pratiques, il est alors intéressant de calculer des versions normalisées de la matrice $\mathbf{R}$:

- $\mathbf{R}_l$ pour laquelle chaque ligne i est divisée par $|\mathbf{r}_{i:}|$,
- $\mathbf{R}_c$ pour laquelle chaque colonne i est divisée par $|\mathbf{r}_{:i}|$.

Le principal problème de cette normalisation est qu'elle ne garantit pas que la similarité d'un document avec lui même soit maximale à 1.

Finalement, à partir de la matrice de similarité entre les mots du vocabulaire $\mathbf{S}_2$, on peut calculer la matrice de similarité entre les documents $\mathbf{S}_1$ comme suit :

$$\mathbf{S}_1 = \mathbf{R}_l \times \mathbf{S}_2 \times \mathbf{R}_l^\top \tag{10a}$$

Comme précisé dans l'introduction de cette section, nous nous sommes concentrés sur le calcul des similarités entre documents. Cependant, on peut utiliser le raisonnement dual pour s'intéresser aux similarités entre mots $\mathbf{S}_2$ que l'on peut calculer à partir des similarités entre documents $\mathbf{S}_1$:

$$\mathbf{S}_2 = \mathbf{R}_c^\top \times \mathbf{S}_1 \times \mathbf{R}_c \tag{10b}$$

Dans le cas général, on ne possède pas une matrice $\mathbf{S}_2$ représentant la vérité absolue sur les similarités entre termes (ni une matrice $\mathbf{S}_1$, sinon on pourrait directement faire la classification). Nous allons donc voir dans la section suivante comment utiliser ces formules pour calculer ces matrices de similarités $\mathbf{S}_1$ et $\mathbf{S}_2$.

II-1.3 Algorithme original

Comme nous l'avons vu dans la partie précédente, si l'on suppose connue la matrice de similarités entre tous les termes du corpus, on peut alors calculer la matrice de similarité entre tous les documents. De la même façon, on peut calculer les similarités entre les mots à partir des similarités entre les documents. Le problème est que l'on ne connaît *a priori* ni l'une ni l'autre (ni $\mathbf{S}_1$, ni $\mathbf{S}_2$), on va donc transformer les Eq. (10) en règles de mise à jour, dans lesquelles t représente l'itération courante :

$$\mathbf{S}_1^{(t)} \leftarrow \mathbf{R}_l \times \mathbf{S}_2^{(t-1)} \times \mathbf{R}_l^\top \tag{11a}$$

$$\mathbf{S}_2^{(t)} \leftarrow \mathbf{R}_c^\top \times \mathbf{S}_1^{(t-1)} \times \mathbf{R}_c \tag{11b}$$

De plus, dans le cas général lorsque l'on ne dispose pas de connaissances *a priori* sur les données, on initialise les matrices $\mathbf{S}_1^{(0)}$ et $\mathbf{S}_2^{(0)}$ avec la matrice identité $\mathbf{I}$. Il est bien sûr possible d'initialiser la matrice de similarité entre mots avec une autre matrice que l'identité, si l'on dispose de similarités à partir d'une ontologie par exemple. Le principe de l'algorithme χ-Sim repose donc sur le fait de calculer alternativement les similarités entre documents à partir des similarités entre termes, les similarités entre termes à partir des similarités entre documents. L'algorithme 1 résume le déroulement de l'algorithme χ-Sim.

Le nombre d'itérations de l'algorithme est noté T, et Bisson et Hussain (2008) précisent qu'il est généralement fixé à 4. En pratique le nombre d'itérations a un sens très fort, car il correspond au degré maximum de co-occurrences d'ordre supérieur explorées. Nous définissons dans la Def. 3 la notion de co-occurrence d'ordre quelconque.

Définition 3. *Une co-occurrence d'ordre u est un chemin de longueur $2\,u$ entre deux objets de même type dans un graphe biparti. Une co-occurrence d'ordre 1 entre deux documents signifie qu'ils partagent des mots communs, et on désignera par co-occurrences d'ordre supérieur les co-occurrences d'ordre supérieur ou égal à 2.*

Algorithme 1 L'algorithme χ-Sim original de Bisson et Hussain (2008).

Entrées : la matrice de relation $\mathbf{R}$, paramètre : nombre d'itérations T
Sorties : les matrices de similarités $\mathbf{S}_1$ et $\mathbf{S_2}$

1: $\mathbf{S}_1^{(0)} \leftarrow \mathbf{I}$
2: $\mathbf{S}_2^{(0)} \leftarrow \mathbf{I}$
3: **Pour** $t = 1 \rightarrow T$ **:**
4: Calcul de $\mathbf{S}_1^{(t)}$ basé sur $\mathbf{S}_2^{(t-1)}$ à partir de (11a)
5: Diagonale de $\mathbf{S}_1^{(t)}$ mise à 1
6: Calcul de $\mathbf{S}_2^{(t)}$ basé sur $\mathbf{S}_1^{(t-1)}$ à partir de (11b)
7: Diagonale de $\mathbf{S}_2^{(t)}$ mise à 1
8: **Fin**

Nous pouvons illustrer la notion de co-occurrence d'ordre supérieur à l'aide d'un exemple simple de corpus représenté par un graphe biparti, comme dans la Fig. 11. On y remarque que les documents $x_1^{(1)}$ et $x_2^{(1)}$ partagent le mot $x_2^{(2)}$, et sont donc liés par une co-occurrence d'ordre 1, ou co-occurrence directe. De plus, on note que les documents $x_1^{(1)}$ et $x_3^{(1)}$ n'ont en commun aucun mot du corpus, mais sont liés par une relation de co-occurrences d'ordre 2, il existe au moins deux façons d'appréhender cette similarité d'ordre supérieure :

1. $x_1^{(1)}$ et $x_3^{(1)}$ sont tous les deux liés à $x_2^{(1)}$ par une co-occurrence d'ordre 1, donc ils sont eux-mêmes liés par une co-occurrence d'ordre 2. Cette vision est illustrée par l'adage *« les amis de mes amis sont mes amis »*.
2. Les mots $x_2^{(2)}$ et $x_3^{(2)}$ sont liés par une relation de co-occurrence d'ordre 1 (ils sont donc similaires), et comme $x_2^{(2)}$ apparaît dans $x_1^{(1)}$ et $x_3^{(2)}$ apparaît dans $x_3^{(1)}$, alors les documents $x_1^{(1)}$ et $x_3^{(1)}$ sont liés par une co-occurrence d'ordre 2.

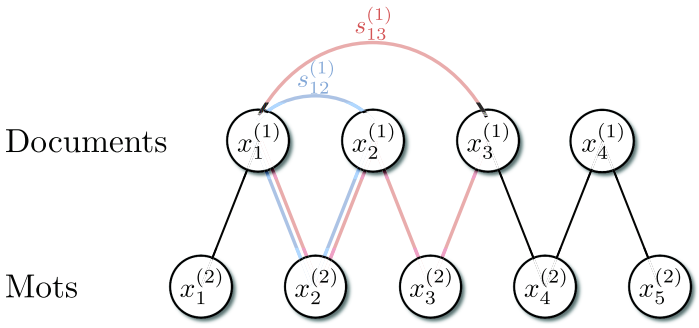


Figure 11 – Illustration de co-occurrences d'ordre supérieurs dans un corpus simple, représenté sous la forme d'un graphe biparti.

Ainsi, chaque itération supplémentaire de l'algorithme χ-Sim revient à prendre en compte des co-occurences d'ordre supérieur plus grandes. Ainsi, à l'itération t, les co-occurrences d'ordre t entre documents (et entre mots), vont contribuer à renforcer les valeurs des similarités entre les objets de la collection. Donc le paramètre T de l'algorithme correspond au degré maximal de co-occurrences que l'on souhaite considérer. Cette notion de degré de co-occurrences a été étudié dans le domaine des sciences cognitives (Lemaire et Denhière 2008), et il a été montré que dans le cas des collections de mots et de documents, des co-occurrences d'ordre supérieurs à 4 étaient peu pertinentes d'un point de vue sémantique.

II-2 Nouvelle normalisation et pseudo-norme

Dans cette section, nous proposons une nouvelle normalisation pour l'algorithme χ-Sim, en partie basée sur la notion de Cosinus généralisé (Qamar et Gaussier 2009). Nous allons ainsi montrer que les matrices de similarités alors calculées par l'algorithme sont semi-définies positives. Nous proposerons par la suite une modification de cette normalisation en introduisant une notion de pseudo-norme, permettant d'améliorer la pertinence des similarités calculées pour des données de grande dimension.

II-2.1 La mesure de similarité du Cosinus généralisé

Nous présentons dans cette section, la mesure de similarité du Cosinus généralisé, introduite par Qamar et Gaussier (2009) dans un contexte d'apprentissage supervisé. Dans un premier temps, nous donnons la définition d'un produit scalaire et d'un semi-produit scalaire dans la Def. 4.

Définition 4. *Soit* $\mathbf{E}$ *un espace vectoriel, l'application* $\langle \cdot, \cdot \rangle$ *:*

$$\begin{aligned} \mathbf{E} \times \mathbf{E} &\longmapsto \mathbb{R} \\ (\mathbf{r}_{i:}, \mathbf{r}_{j:}) &\longmapsto \langle \mathbf{r}_{i:}, \mathbf{r}_{j:} \rangle \end{aligned}$$

est un semi-produit scalaire si elle est :

- *bilinéaire : linéaire par rapport à chaque argument quand l'autre est fixé.*
- *symétrique :* $\forall \mathbf{r}_{i:}, \mathbf{r}_{j:} \in \mathbf{E}^2, \langle \mathbf{r}_{i:}, \mathbf{r}_{j:} \rangle = \langle \mathbf{r}_{j:}, \mathbf{r}_{i:} \rangle$
- *positive :* $\forall \mathbf{r}_{i:} \in \mathbf{E}, \langle \mathbf{r}_{i:}, \mathbf{r}_{i:} \rangle \geqslant 0$

De plus, cette application est un produit scalaire si elle est :

- *définie :* $\langle \mathbf{r}_{i:}, \mathbf{r}_{i:} \rangle = 0 \Rightarrow \mathbf{r}_{i:} = 0$

On peut alors associer au produit scalaire (respectivement au semi-produit scalaire), la norme (respectivement la semi-norme) suivante : $\forall \mathbf{r}_{i:} \in \mathbf{E}, \|\mathbf{r}_{i:}\| = \sqrt{\langle \mathbf{r}_{i:}, \mathbf{r}_{i:} \rangle}$.

La Def. 4 définit les notions de semi-produits scalaires et de produits scalaires dans un espace vectoriel $\mathbf{E}$ quelconque. On introduit à présent la famille de (semi-)produits scalaires suivantes :

Définition 5. *Soit* $\mathbf{S}$ *une matrice semi-définie positive, alors l'application :*

$$\begin{aligned} \mathbf{E} \times \mathbf{E} &\longmapsto \mathbb{R} \\ (\mathbf{r}_{i:}, \mathbf{r}_{j:}) &\longmapsto \mathbf{r}_{i:} \times \mathbf{S} \times \mathbf{r}_{j:}^{\top} \end{aligned}$$

est un semi-produit scalaire, que l'on note $\langle \mathbf{r}_{i:}, \mathbf{r}_{j:} \rangle_{\mathbf{S}}$. *Si la matrice* $\mathbf{S}$ *est définie positive, alors cette application est un produit scalaire. On note* $\|\mathbf{r}_{i:}\|_{\mathbf{S}}$ *la norme (ou semi-norme) associée.*

On sait de plus par la factorisation de Cholesky, que pour toute matrice $\mathbf{S}$ symétrique SDP, il existe une matrice $\mathbf{L}$ triangulaire inférieure telle que $\mathbf{S} = \mathbf{L} \times \mathbf{L}^{\top}$. On peut ainsi ré-écrire le semi-produit scalaire $\langle \mathbf{r}_{i:}, \mathbf{r}_{j:} \rangle_{\mathbf{S}} = \mathbf{r}_{i:} \times \mathbf{S} \times \mathbf{r}_{j:}^{\top} = (\mathbf{r}_{i:}\, \mathbf{L}) \times (\mathbf{r}_{j:}\, \mathbf{L})^{\top}$. On remarque que $\forall \mathbf{r}_{i:} \in \mathbf{E}$, $(\mathbf{r}_{i:}\, \mathbf{L})$ est l'image du vecteur $\mathbf{r}_{i:}$ dans la base induite par $\mathbf{L}$, et donc $\langle \cdot, \cdot \rangle_{\mathbf{S}}$ est le produit-scalaire dans l'espace induit par cette base.

En utilisant la notion de semi-produit scalaire et les notations de la Def. 5, nous présentons dans la Def. 6 la mesure de similarité du Cosinus généralisé, qui a été proposée par Qamar et Gaussier (2009).

Définition 6. *Soit* $\mathbf{E}$ *un espace vectoriel et* $\mathbf{S}$ *une matrice SDP, alors* $\forall \mathbf{r}_{i:},\mathbf{r}_{j:} \in \mathbf{E}^2$ *tels que* $\|\mathbf{r}_{i:}\|_{\mathbf{S}} \neq 0$ *et* $\|\mathbf{r}_{j:}\|_{\mathbf{S}} \neq 0$*, la similarité du Cosinus généralisée entre* $\mathbf{r}_{i:}$ *et* $\mathbf{r}_{j:}$ *s'écrit :*

$$\mathrm{Cos}_{\mathbf{S}}(\mathbf{r}_{i:},\mathbf{r}_{j:}) = \frac{\langle \mathbf{r}_{i:},\mathbf{r}_{j:} \rangle_{\mathbf{S}}}{\|\mathbf{r}_{i:}\|_{\mathbf{S}} \times \|\mathbf{r}_{j:}\|_{\mathbf{S}}} \tag{12}$$

A l'instar de notre remarque sur le semi-produit scalaire $\langle \mathbf{r}_{i:},\mathbf{r}_{j:} \rangle_{\mathbf{S}}$, on peut affirmer que cette mesure de similarité correspond à la mesure du Cosinus dans la base induite par $\mathbf{L}$ (la mesure du Cosinus classique étant dans la base canonique, quand $\mathbf{S}$ est la matrice identité). Au regard de sa définition, nous confirmons que la valeur de cette mesure de similarité est bien comprise dans l'intervalle [0,1], et que $\forall \mathbf{r}_{i:} \in \mathbf{E}$, $\mathrm{Cos}_{\mathbf{S}}(\mathbf{r}_{i:},\mathbf{r}_{i:}) = 1$.

Pour simplifier les notations dans la suite de ce manuscrit, nous introduisons une notation pour la matrice dont le terme général est la mesure du Cosinus généralisée de deux vecteurs dans la Def. 7.

Définition 7. *Pour toute matrice* $\mathbf{S}$ *SDP de taille* $\eta_2 \times \eta_2$*, et pour toute matrice* $\mathbf{R}$*, de taille* $\eta_1 \times \eta_2$*, telle que* $\forall i \in [\![1,\eta_1]\!]$, $\|\mathbf{r}_{i:}\|_{\mathbf{S}} > 0$*, on définit la fonction* $\mathrm{Cos}_{\mathbf{S}}(\mathbf{R}) = \mathbf{C}$*, telle que* $\mathbf{C} = (c_{ij})_{1 \leqslant i,j \leqslant \eta_1}$ *avec* $c_{ij} = \mathrm{Cos}_{\mathbf{S}}(\mathbf{r}_{i:},\mathbf{r}_{j:})$ *(voir l'Eq. (12)).*

La Prop. 1 affirme maintenant que la matrice $\mathrm{Cos}_{\mathbf{S}}(\mathbf{R})$ définie dans la Def. 7 est semi-définie positive.

Proposition 1. *Soit* $\mathbf{S}$ *une matrice symétrique SDP de taille* $\eta_2 \times \eta_2$*, et* $\mathbf{R}$ *une matrice de taille* $\eta_1 \times \eta_2$*, telle que* $\forall i \in [\![1,\eta_1]\!]$, $\|\mathbf{r}_{i:}\|_{\mathbf{S}} > 0$*, alors la matrice* $\mathbf{C} = \mathrm{Cos}_{\mathbf{S}}(\mathbf{R})$ *est également SDP.*

Démonstration. $\forall i \in [\![1,\eta_1]\!]$, on peut normaliser la ligne i de la matrice $\mathbf{R}$ par sa norme définie par la matrice SDP $\mathbf{S}$: $\mathbf{r}_{i:}$ devient alors $\hat{\mathbf{r}_{i:}} = \mathbf{r}_{i:}/(\|\mathbf{r}_{i:}\|_{\mathbf{S}})$. On note alors $\hat{\mathbf{R}}$ la matrice $\mathbf{R}$ dont les lignes sont ainsi normalisées. La matrice $\mathbf{S}$ étant symétrique SDP, la factorisation de Cholesky nous assure qu'il existe une matrice $\mathbf{L}$ triangulaire inférieure telle que $\mathbf{S} = \mathbf{L} \times \mathbf{L}^\top$, on peut donc écrire $\mathrm{Cos}_{\mathbf{S}}(\mathbf{R}) = \hat{\mathbf{R}} \times \mathbf{S} \times \hat{\mathbf{R}}^\top = (\hat{\mathbf{R}}\,\mathbf{L}) \times (\hat{\mathbf{R}}\,\mathbf{L})^\top$. Pour montrer que la matrice $\mathrm{Cos}_{\mathbf{S}}(\mathbf{R})$ est SDP, il faut et il suffit de prouver que $\forall \mathbf{x} \in \mathbb{R}^{\eta_1}$ on a :

$$\begin{aligned} & \mathbf{x} \times \mathrm{Cos}_{\mathbf{S}}(\mathbf{R}) \times \mathbf{x}^\top && \geqslant 0 \\ \Leftrightarrow \quad & \mathbf{x} \times (\hat{\mathbf{R}}\,\mathbf{L}) \times (\hat{\mathbf{R}}\,\mathbf{L})^\top \times \mathbf{x}^\top && \geqslant 0 \\ \Leftrightarrow \quad & (\mathbf{x}\,\hat{\mathbf{R}}\,\mathbf{L}) \times (\mathbf{x}\,\hat{\mathbf{R}}\,\mathbf{L})^\top && \geqslant 0 \end{aligned} \tag{13}$$

En notant $\mathbf{a} = \mathbf{x}\,\hat{\mathbf{R}}\,\mathbf{L}$ le vecteur de terme général a_i, on obtient $(\mathbf{x}\,\hat{\mathbf{R}}\,\mathbf{L}) \times (\mathbf{x}\,\hat{\mathbf{R}}\,\mathbf{L})^\top = \sum_{i=1}^{\eta_2} (a_i)^2 \geqslant 0$. □

Maintenant que nous avons établi que la matrice composée de toutes les mesures de similarité du Cosinus généralisé était SDP, nous allons adapter les Eq. (11) définissant les règles de mise à jour de l'algorithme χ-Sim original, afin d'obtenir une nouvelle version de celui-ci. Nous définissons alors une suite de matrices de similarités dans la Def. 8, utilisant la notion de matrice du Cosinus généralisé de la Def. 7.

Définition 8. *Soit* $\mathbf{R}$ *une matrice quelconque de taille* $\eta_1 \times \eta_2$ *à termes positifs et sans ligne ou colonne nulle, alors nous définissons les deux suites de matrices suivantes :*

$$\begin{cases} \mathbf{S}_1^{(0)} & = \mathbf{I}_{\eta_1} \text{ et } \mathbf{S}_2^{(0)} = \mathbf{I}_{\eta_2} \\ \mathbf{S}_1^{(t)} & = \mathrm{Cos}_{\mathbf{S}_2^{(t-1)}}(\mathbf{R}) \\ \mathbf{S}_2^{(t)} & = \mathrm{Cos}_{\mathbf{S}_1^{(t-1)}}(\mathbf{R}^\top) \end{cases}$$

Nous allons maintenant montrer que ces suites de matrices de similarités existent, c'est à dire que la semi-norme des lignes et des colonnes de $\mathbf{R}$ induites par ces matrices de similarités est toujours strictement positive, et que de plus toutes ces matrices sont SDP.

Proposition 2. *Les deux suites de matrices de similarités* $\mathbf{S}_1^{(t)}$ *et* $\mathbf{S}_2^{(t)}$ *définies dans la Def. 8 existent et sont SDP pour tout* t*.*

Démonstration. Par récurrence :

- $\mathbf{S}_1^{(0)}$ et $\mathbf{S}_2^{(0)}$ existent et sont SDP car la matrice identité est SDP.
- Supposons que $\mathbf{S}_2^{(t-1)}$, notée $\mathbf{S}$ pour simplifier les notations, existe et est SDP. Alors $\mathbf{S}_1^{(t)}$ existe si $\forall i \in [\![1,\eta_1]\!] : \|\mathbf{r}_{i:}\|_{\mathbf{S}} > 0$. Or :

$$\begin{aligned} \|\mathbf{r}_{i:}\|_{\mathbf{S}} &= \mathbf{r}_{i:} \times \mathbf{S} \times \mathbf{r}_{i:}^\top \\ &= \sum_j^{\eta_2} \sum_k^{\eta_2} r_{ij}\, r_{ik}\, s_{jk} \\ &\geqslant \sum_j^{\eta_2} r_{ij}^2 \qquad \text{car } s_{jj} = 1 \text{ et } r_{ij} \geqslant 0 \\ &> 0 \end{aligned}$$

  Finalement, en utilisant la proposition 1, on obtient que $\mathbf{S}_1^{(t)}$ est SDP. La démonstration est tout à fait similaire pour montrer que si $\mathbf{S}_1^{(t-1)}$ existe et est SDP, alors $\mathbf{S}_2^{(t)}$ existe et est SDP également.

□

Finalement, nous avons démontré que les matrices de similarité calculées à l'aide de la fonction du Cosinus généralisé de la Def. 6, sont semi-définies positives, c'est à dire que toutes leurs valeurs propres sont positives ou nulles. Notons $q_1^{(t)}$ (respectivement $q_2^{(t)}$) le nombre de valeurs propres strictement positives de $\mathbf{S}_1^{(t)}$ (respectivement $\mathbf{S}_2^{(t)}$), et q le rang de $\mathbf{R}$. Or comme l'algorithme calcule les matrices de similarités comme un produit matriciel faisant intervenir $\mathbf{R}$ et les matrices de similarités de l'itération précédente, nous pouvons affirmer que $\forall t, q_1^{(t+1)} \leqslant q_1^{(t)} \leqslant q$ et $q_2^{(t+1)} \leqslant q_2^{(t)} \leqslant q$. Donc le nombre de valeurs propres strictement

positives de ces matrices de similarité décroît et atteint nécessairement un minimum. Si ce minimum est 1, alors ces matrices sont des matrices unités (matrices de 1). Ceci justifie le fait que Bisson et Hussain (2008) fixent généralement le nombre d'itérations T de χ-Sim à 4.

II-2.2 Introduction d'une pseudo-norme

Nous avons défini dans l'Eq. (1) dans la Sec. I-2.1 la distance de Minkowski. Ajoutons qu'il est possible de lui associer une norme, classiquement appelée norme L_k dans la littérature et notée $\|\cdot\|_k$ qui représente la distance de Minkowski d'un vecteur à l'origine :

$$\|\mathbf{r}_{i:}\|_k = \left(\sum_{l=1}^{\eta_1} (r_{il})^k\right)^{\frac{1}{k}}$$

Aggarwal *et al.* (2001) ont étudié le comportement de cette norme classique dans les espaces à très grandes dimensions, et ont observé un comportement intéressant, que nous avons déjà évoqué dans la Sec. I-2.1.c sur le fléau de la dimensionnalité. Les auteurs montrent que ce qu'ils dénomment *contraste relatif*, et qui est représenté par le ratio $\frac{\mathrm{D}_{max}-\mathrm{D}_{min}}{\mathrm{D}_{min}}$ avec D_{max} et D_{min} représentant respectivement la norme L_k du plus grand et du plus petit vecteur de l'espace, tend vers 0 quand le nombre de dimensions tend vers $+\infty$, et ce pour une distribution aléatoire de points.

Rappelons que dans le modèle vectoriel, un document est décrit par un vecteur de taille égale au nombre de mots présents dans le corpus η_2, avec η_2 généralement grand, et donc la mesure de co-similarité χ-Sim travaille dans des espaces à grandes dimensions. Nous nous inspirons de l'observation faite par Aggarwal *et al.* (2001) pour définir alors la notion de pseudo produit scalaire et de pseudo-norme, de façon similaire à l'Eq. (12) :

Définition 9. *Pour toute matrice* $\mathbf{S}$ *et* $k \in \mathbb{R}^*$*, on peut définir l'application et la pseudo-norme associée suivantes :*

$$\langle \mathbf{r}_{i:},\mathbf{r}_{j:} \rangle_{\mathbf{S}}^k = \left((\mathbf{r}_{i:})^k \times \mathbf{S} \times (\mathbf{r}_{j:}^\top)^k\right)^{\frac{1}{k}} \qquad et \qquad \|\mathbf{r}_{i:}\|_{\mathbf{S}}^k = \langle \mathbf{r}_{i:},\mathbf{r}_{i:} \rangle_{\mathbf{S}}^k \tag{14}$$

en définissant $(\mathbf{r}_{i:})^k$ *comme la mise à la puissance* k *des termes du vecteur* $\mathbf{r}_{i:}$*. Nous introduisons alors la pseudo similarité du Cosinus généralisée, qui s'écrit alors :*

$$\mathrm{Cos}_{\mathbf{S}}^k(\mathbf{r}_{i:},\mathbf{r}_{j:}) = \frac{\langle \mathbf{r}_{i:},\mathbf{r}_{j:} \rangle_{\mathbf{S}}^k}{\sqrt{\|\mathbf{r}_{i:}\|_{\mathbf{S}}^k \times \|\mathbf{r}_{j:}\|_{\mathbf{S}}^k}} \tag{15}$$

On ne peut pas parler de produit scalaire pour l'application $\langle \cdot\,,\cdot \rangle_{\mathbf{S}}^k$, ni même de semi-produit scalaire, car cette application n'est pas bilinéaire. De la même façon pour le Cosinus généralisé, nous introduisons une notation pour la matrice dont le terme général est la pseudo similarité du Cosinus généralisée de deux vecteurs.

Définition 10. *Pour toute matrice* $\mathbf{S}$ *de taille* $\eta_2 \times \eta_2$*, et pour toute matrice* $\mathbf{R}$ *quelconque, de taille* $\eta_1 \times \eta_2$*, telle que* $\forall i \in [\![1,\eta_1]\!]$, $\|\mathbf{r}_{i:}\|_{\mathbf{S}}^k > 0$*, on définit la fonction* $\mathrm{Cos}_{\mathbf{S}}^k(\mathbf{R}) = \mathbf{C}$*, telle que* $\mathbf{C} = (c_{ij})_{1 \leqslant i,j \leqslant \eta_1}$ *avec* $c_{ij} = \mathrm{Cos}_{\mathbf{S}}^k(\mathbf{r}_{i:},\mathbf{r}_{j:})$ *(voir l'Eq. (15)).*

En utilisant ainsi le même schéma de normalisation que la mesure de similarité du Cosinus (et du Cosinus généralisé), nous avons la garantie que la similarité entre un élément et lui même est de 1, et que les autres similarités sont comprises dans l'intervalle [0,1].

II-2.3 Nouvelle version de l'algorithme χ-Sim

L'algorithme 2 présente la nouvelle version de l'algorithme χ-Sim, utilisant cette nouvelle normalisation et la pseudo-norme k. Pour $k = 1$, la proposition 1 nous assure que les matrices de similarités calculées par cet algorithme sont toutes SDP. Cependant, nous ne pouvons apporter aucune garantie théorique pour des valeurs de k différentes de 1. En revanche, nous espérons observer une amélioration des performances expérimentales, comme nous le verrons dans le Chap. IV.

Algorithme 2 L'algorithme χ-Sim utilisant le pseudo Cosinus généralisé.

Entrées : la matrice de relations $\mathbf{R}$, paramètres : nombre d'itération T, pseudo-norme k
Sorties : les matrices de similarités $\mathbf{S}_1$ et $\mathbf{S_2}$

1: $\mathbf{S}_1^{(0)} \leftarrow \mathbf{I}$
2: $\mathbf{S}_2^{(0)} \leftarrow \mathbf{I}$
3: **Pour** $t = 1 \rightarrow T$ **:**
4: $\quad \mathbf{S}_1^{(t)} \leftarrow \mathrm{Cos}^k_{\mathbf{S}_2^{(t-1)}}(\mathbf{R})$
5: $\quad \mathbf{S}_2^{(t)} \leftarrow \mathrm{Cos}^k_{\mathbf{S}_1^{(t-1)}}(\mathbf{R}^\top)$
6: **Fin**

D'un point de vue pratique, il est possible d'expliciter le calcul de $\mathrm{Cos}^k_{\mathbf{S}}(\mathbf{R})$ de la façon suivante :

$$\mathbf{P} \leftarrow \left(\mathbf{R}^{\circ_k} \times \mathbf{S}_2^{(t-1)} \times (\mathbf{R}^{\circ_k})^\top\right)^{\frac{1}{k}} \tag{16a}$$

$$\mathrm{Cos}^k_{\mathbf{S}}(\mathbf{R}) \leftarrow \frac{\mathbf{P}}{\sqrt{\mathrm{diag}\,(\mathbf{P}) \otimes \mathrm{diag}\,(\mathbf{P})}} \tag{16b}$$

avec les notations suivantes :

- $\mathbf{R}^{\circ_k}$ désigne la mise à puissance k des termes d'une matrice $\mathbf{R}$,
- $\mathrm{diag}\,(\mathbf{P})$ désigne le vecteur composé des termes diagonaux de la matrice $\mathbf{P}$,
- le symbole $\otimes$ désigne le produit tensoriel entre deux vecteurs, dont le résultat est une matrice,
- et la division de deux matrices de même taille définie comme la division terme à terme.

Terminons cette section par une remarque sur la complexité de l'algorithme χ-Sim (qui reste inchangée avec la nouvelle version), aussi bien temporelle que spatiale, en faisant l'hypothèse que $\eta_1 = \eta_2 = \eta$. La complexité temporelle du produit de matrices est alors en $\mathcal{O}(\eta^3)$, et la normalisation en $\mathcal{O}(\eta^2)$, ce qui donne une complexité temporelle totale (on suppose le nombre d'itérations négligeable devant η) de $\mathcal{O}(\eta^3)$. Quant à la complexité spatiale, elle est principalement due au stockage nécessaire des deux matrices de similarité, et est donc en

$\mathcal{O}(\eta^2)$. Notons que la matrice de relation $\mathbf{R}$ est dans la plupart des applications creuse, alors que les matrices de similarités ne le sont généralement pas, en particulier celles calculées par l'algorithme χ-Sim.

II-3 Étude du graphe des documents

Nous avons vu un exemple de graphe bipartite dans la Fig. 11 de la Sec. II-1, mais dans cette section, nous allons nous intéresser à un autre graphe utile pour décrire de telles données, plus synthétique car on y omet les informations relatives au mots : c'est le graphe des documents. Ce graphe est très simple à construire car sa matrice d'adjacence est simplement obtenue, pour une matrice documents - mots $\mathbf{R}$ en calculant $\mathbf{R} \times \mathbf{R}^\top$. En partant d'un graphe bipartite document - mots, il suffit de ne garder que les sommets associés à des documents, et d'ajouter une arête entre deux documents s'ils partagent au moins un mot. Le poids de cette arête étant dépendant du modèle de pondération choisi pour $\mathbf{R}$ (voir Sec. I-1.1.b). Dans le cas du modèle binaire, le poids des arêtes entre deux sommets sera simplement le nombre de mots en commun entre ces deux documents. La Fig. 12 montre un exemple de ce type de graphe, associé au graphe bipartite présenté dans la Fig. 11.

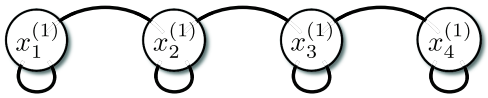


Figure 12 – Le graphe des documents associé au graphe bipartite de la Fig. 11.

Les éléments de la matrice $\mathbf{S}_1$ après la première itération de χ-Sim correspondent aux chemins d'ordre 1 (ou chemins directs) pondérés dans le graphe des documents. En effet après la première itération, $s_{ij}^{(1)}$ représente alors le nombre de chemins directs, i.e. le nombre de mots en commun, entre les documents i et j. A la normalisation près, $\mathbf{S}_1$ après la première itération est la matrice d'adjacence du graphe des documents. Rappelons que plus généralement l'itération t de l'algorithme χ-Sim (dans sa version originale ou dans sa nouvelle version) revient à prendre en compte des chemins de longueurs t dans ce graphe. Considérons ainsi la matrice de similarité entre les documents de la Fig. 12 présentée dans la Fig. 13, dans laquelle les termes deviennent tous non-nuls après 3 itérations de l'algorithme.

	$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$	$x_4^{(1)}$
$x_1^{(1)}$	1	$s_{12}^{(1)}$	$s_{13}^{(1)}$	$s_{14}^{(1)}$
$x_2^{(1)}$	$s_{21}^{(1)}$	1	$s_{23}^{(1)}$	$s_{24}^{(1)}$
$x_3^{(1)}$	$s_{31}^{(1)}$	$s_{32}^{(1)}$	1	$s_{34}^{(1)}$
$x_4^{(1)}$	$s_{41}^{(1)}$	$s_{42}^{(1)}$	$s_{43}^{(1)}$	1

Figure 13 – Matrice de similarité entre les documents de la Fig. 12. Les termes $s_{12}^{(1)}$, $s_{23}^{(1)}$ et $s_{34}^{(1)}$ deviendront non-nuls à l'itération 1 ; puis les termes $s_{13}^{(1)}$ et $s_{24}^{(1)}$ deviendront non-nuls à l'itération 2 ; et finalement le terme $s_{14}^{(1)}$ deviendra non-nul à l'itération 3.

Au regard de l'analyse du comportement de l'algorithme χ-Sim, on peut détecter trois problèmes, que nous allons détailler dans la suite de cette section :

- Les similarités (même faibles) entre mots induisant des similarités entre documents et *vice versa*, si une similarité est sur-évaluée, cette erreur va se propager.
- Les chemins redondants (chemins qui passent plusieurs fois par un même sommet) entre les documents sont indifféremment pris en compte.
- La longueur des chemins n'est pas prise en compte (un chemin de longueur 3 aura la même importance qu'un chemin de longueur 1).

II-3.1 Le seuillage : une solution contre la propagation du bruit

Dans un corpus de documents, nous pouvons observer que pour un document donné, un certain nombre de mots, bien que présents, n'apparaissent qu'avec un nombre faible d'occurrences, et ne sont pas toujours pertinents par rapport à la classe de ce document. Pour être plus précis, ces mots sont peut-être utiles dans le corpus, car ils peuvent être caractéristiques d'une certaine classe, mais leur présence dans des documents d'autres classes est nuisible car elle va créer des similarités entre des documents de classes différentes.

L'hypothèse que nous faisons alors est que ces similarités entre des documents de classes différentes sont faibles, et donc que toutes les valeurs de similarités faibles ne sont pas pertinentes et peuvent être identifiées à du bruit. Le principal problème causé par la présence d'un tel bruit dans les matrices de similarité est que de part l'intuition de base de χ-Sim (les similarités entre mots sont utilisées pour calculer les similarités entre documents), ce bruit va avoir tendance à se propager et à s'amplifier au fur et à mesure des itérations de l'algorithme.

Partant de cette hypothèse, la solution que nous proposons est d'ajouter une étape de seuillage à chaque itération de l'algorithme χ-Sim. Cette étape consiste à annuler $p\%$ des plus faibles valeurs de similarités. En reprenant l'algorithme 2, il suffit d'ajouter un paramètre de seuillage p puis d'ajouter après la ligne 4 (respectivement la ligne 5) cette étape de seuillage appliquée à la matrice de similarité $\mathbf{S}_1^{(t)}$ (respectivement à $\mathbf{S}_2^{(t)}$).

> ***Remarque 4.*** *Cette étape de seuillage peut également être ajoutée à l'algorithme original de χ-Sim de Bisson et Hussain (2008).*

Il est important de remarquer que cette étape ne modifie ni la complexité temporelle, ni la complexité spatiale de l'algorithme χ-Sim. En effet, pour calculer la complexité temporelle de l'étape de seuillage, il faut considérer le tri des valeurs de similarités qui sont de l'ordre de η^2 (on considère toujours une matrice de relation $\mathbf{R}$ de taille générale $\eta \times \eta$). Les algorithmes de tri les plus rapides ont une complexité temporelle en $\mathcal{O}(n' \log(n'))$ pour n' valeurs à trier. Ainsi, la complexité de cette étape sera en $\mathcal{O}(\eta^2 \log(\eta^2))$, et comme la complexité de χ-Sim est en $\mathcal{O}(\eta^3)$, elle reste inchangée. Quant à la complexité spatiale, elle reste également en $\mathcal{O}(\eta^2)$.

II-3.2 Les chemins redondants et l'amortissement

Jusqu'à présent nous avons expliqué que l'itération t de l'algorithme χ-Sim revient à prendre en compte les chemins de longueurs t dans le graphe des documents (la même explication étant valable pour les mots). Cependant, nous pouvons classer ces chemins en deux catégories :

- les chemins non-redondants : chemins qui ne passent pas deux fois par un même sommet,
- les chemin redondants : chemins qui passent au moins deux fois par l'un des sommets.

En reprenant l'exemple du graphe de documents de la Fig. 12 et de la matrice de similarité entre ces documents de la Fig. 13, on peut par exemple considérer la similarité entre les documents $x_1^{(1)}$ et $x_2^{(1)}$ notée $s_{12}^{(1)}$. Comme nous l'avons vu, ces documents étant liés par un chemin direct, cette similarité deviendra non-nulle dès la première itération de l'algorithme. Cependant, il existe également de nombreux chemins redondants entre ces documents qui vont donc modifier cette similarité au fil des itérations de χ-Sim, dont voilà des exemples de différentes longueurs :

- de longueur 2 : $x_1^{(1)} \rightarrow x_1^{(1)} \rightarrow x_2^{(1)}$,
- de longueur 3 : $x_1^{(1)} \rightarrow x_2^{(1)} \rightarrow x_3^{(1)} \rightarrow x_2^{(1)}$,
- de longueur 4 : $x_2^{(1)} \rightarrow x_3^{(1)} \rightarrow x_2^{(1)} \rightarrow x_1^{(1)} \rightarrow x_1^{(1)}$.

Il serait intéressant de prendre en compte uniquement les chemins non-redondants car il semble que ce soit ceux qui apportent le plus d'information quant à la similarité entre les sommets du graphe. Malheureusement, il n'existe à notre connaissance aucune méthode permettant de calculer uniquement ces chemins, en éliminant les chemins redondants, pour une taille quelconque (expression analytique pour les chemins non-redondants de longueur 1 et 2).

Ajoutons à ce problème de la prise en compte des chemins redondants, un second défaut de l'approche que nous avons déjà évoqué dans l'introduction de la Sec. II-3 : l'algorithme χ-Sim ne prend pas en compte la longueur des chemins. Nous proposons d'introduire un facteur d'amortissement pour que plus un chemin soit long, plus son influence soit restreinte. Comme nous l'avons précédemment dit la matrice $\mathbf{A} = \mathbf{R}\,\mathbf{R}^\top$ (matrice d'adjacence du graphe des documents) est équivalente aux chemins d'ordre 1 (ou directs), à la normalisation près. De plus, nous affirmons que la matrice $\mathbf{A}\,\mathbf{A} = \mathbf{A}^2$ représente de la même façon les chemins d'ordre 2. De façon générale, la matrice $\mathbf{A}^t$ représente les chemins d'ordre t, redondants ou pas. On dispose ainsi d'une suite de matrices $\{\mathbf{A}^t\}$, et l'on souhaite les pondérer avant d'en faire la somme. L'intérêt d'une telle approche serait de ne plus avoir à stopper l'algorithme après 4 itérations, mais de pouvoir le laisser converger vers des valeurs de similarité pertinentes.

Il existe alors plusieurs possibilités pour pondérer ces matrices afin d'amortir les chemins en fonction de leur longueur, et nous en présentons deux :

- soit $\lambda \in]0,1]$ un paramètre d'amortissement, $\sum_{t=1}^{T} \lambda^{t-1}\,\mathbf{A}^t$
- $\sum_{t=1}^{T} \frac{1}{t\,!}\,\mathbf{A}^t \xrightarrow[T\rightarrow+\infty]{} e^{\mathbf{A}}$

La première solution converge si et seulement si λ est inférieur ou égal à l'inverse de la plus grande valeur propre de la matrice $\mathbf{A}$. Et la seconde solution converge systématiquement car cela correspond à la définition de l'exponentielle d'une matrice. Dans les deux cas présentés, se pose cependant le problème de la normalisation des valeurs, car même si l'exponentielle de la matrice $\mathbf{A}$ existe toujours, ses valeurs seront *a priori* très grandes, il convient donc de normaliser ces similarités entre 0 et 1.

> ***Remarque 5.*** *La mesure de similarité* CTK *que nous avons présentée dans la Sec. I-2.3.d se base sur le temps moyen que met un marcheur aléatoire se déplaçant sur le graphe des documents, pour faire l'aller-retour entre deux sommets du graphe. Dans cette approche, les chemins redondants ne sont pas élimés, mais au contraire, ils renforcent les similarités entre les sommets avec l'interprétation que pour aller d'un sommet $x_i^{(1)}$ à un sommet $x_j^{(1)}$, le chemin le plus court est important mais pas uniquement. Si de nombreux chemins redondants permettent de joindre $x_i^{(1)}$ et $x_j^{(1)}$, alors ces deux sommets seront facilement joignables par un marcheur aléatoire, donc la similarité entre ces deux sommets doit être renforcée.*

Finalement, il est possible que le problème des chemins redondants permettent d'introduire un amortissement implicite des chemins en fonction de leur longueur. En fait, les chemins redondants renforcent les chemins les plus courts, et du fait de la normalisation des valeurs de similarité dans l'intervalle [0,1], on obtient un amortissement des chemins les plus longs. La conclusion de cette observation est qu'il n'est pas intéressant d'introduire un tel amortissement s'il n'est pas possible de prendre uniquement en compte les chemin non-redondants.

II-4 Problème d'optimisation lié à χ-Sim

Toujours dans l'objectif de chercher à mieux comprendre le comportement de l'algorithme χ-Sim, et d'essayer de l'améliorer, nous nous sommes proposés de l'exprimer comme un problème d'optimisation sous contraintes. Si l'on considère l'idée de départ de χ-Sim :

- deux documents sont similaires s'ils contiennent des mots similaires,
- et deux mots sont similaires s'ils apparaissent dans des documents similaires.

Il semble que la façon la plus naturelle d'implémenter cette idée de base est de dire qu'à partir de la matrice documents - mots $\mathbf{R}$, nous cherchons une matrice de similarité entre documents $\mathbf{S}_1$ et une matrice de similarité entre mots $\mathbf{S}_2$ telles que :

$$\begin{cases} \mathbf{S}_1 = \mathbf{R} \times \mathbf{S}_2 \times \mathbf{R}^\top \\ \mathbf{S}_2 = \mathbf{R}^\top \times \mathbf{S}_1 \times \mathbf{R} \end{cases} \tag{17}$$

en y ajoutant les contraintes suivantes : auto-similarité de 1, similarités comprises dans l'intervalle [0,1] et symétrie. Ainsi, on considère que les auto-similarités ne sont plus des variables, et la symétrie divise par 2 le nombre de variables recherchées que nous notons $\eta = \frac{\eta_1^2-\eta_1}{2} + \frac{\eta_2^2-\eta_2}{2}$. La solution que nous recherchons est alors un vecteur $\mathbf{s} \in \mathbb{R}^\eta$.

Finalement, nous définissons le problème d'optimisation sous contraintes $(\mathcal{P})$ suivant :

$$\begin{aligned}
\underset{\mathbf{s}\in\mathbb{R}^\eta}{\text{Minimiser}} \quad & J(\mathbf{s}) = \|\mathbf{S}_1 - \mathbf{R}\times\mathbf{S}_2\times\mathbf{R}^\top\|^2 \quad + \quad \|\mathbf{S}_2 - \mathbf{R}^\top\times\mathbf{S}_1\times\mathbf{R}\|^2 \\
\text{avec} \quad & g^1_{ij}(\mathbf{s}) = -s^1_{ij} \leqslant 0 && 1\leqslant i,j\leqslant\eta_1, i<j \\
& g^2_{ij}(\mathbf{s}) = s^1_{ij} - 1 \leqslant 0 && 1\leqslant i,j\leqslant\eta_1, i<j \\
& g^3_{kl}(\mathbf{s}) = -s^2_{kl} \leqslant 0 && 1\leqslant k,l\leqslant\eta_2, k<l \\
& g^4_{kl}(\mathbf{s}) = s^2_{kl} - 1 \leqslant 0 && 1\leqslant k,l\leqslant\eta_2, k<l
\end{aligned}$$

L'ensemble des contraintes (points admissibles) est le suivant :

$$\mathcal{C} = \{\mathbf{s}\in\mathbb{R}^\eta \quad / \quad \forall p\in[\![1,\eta]\!], 0\leqslant s_p\leqslant 1\}$$

On s'intéresse maintenant à déterminer si ce problème admet une solution, et si oui, si elle est unique. La fonction J est continue et de plus $\mathcal{C}$ est fermé comme intersection d'images réciproques de fermés par des fonctions continues (les fonctions g). L'ensemble des contraintes $\mathcal{C}$ étant également borné, on sait que $(\mathcal{P})$ admet au moins une solution.

Finalement, les fonctions g étant convexes, l'ensemble $\mathcal{C}$ est également convexe, et alors il faut et il suffit que la fonction J soit strictement convexe pour que $(\mathcal{P})$ le soit également, et n'admette ainsi qu'au plus une solution.

Proposition 3. *La fonction J définie dans le problème d'optimisation $(\mathcal{P})$ est convexe (mais pas strictement), i.e. :*

$$\forall\mathbf{s},\mathbf{s}'\in\mathbb{R}^\eta \text{ et } \forall t\in[0,1],\ J(t\,\mathbf{s}+(1-t)\,\mathbf{s}')\leqslant t\,J(\mathbf{s})+(1-t)\,J(\mathbf{s}') \tag{18}$$

Démonstration. La preuve de cette proposition se trouve dans l'Annexe A. □

J n'étant pas strictement convexe, le problème $(\mathcal{P})$ admet au moins une solution, mais elle n'est pas nécessairement unique.

Finalement, afin de déterminer s'il est réellement intéressant de chercher à résoudre ce problème, nous avons décidé de tester pour un jeu de données réel quelconque, l'algorithme χ-Sim avec différents paramètres. Pour chacun de ces jeux de paramètres, nous obtenons donc les deux matrices de similarité $\mathbf{S}_1$ et $\mathbf{S}_2$. A partir de la matrice $\mathbf{S}_1$, nous effectuons alors une classification des instances de X_1. Ainsi, nous calculons les deux quantités suivantes :

- la valeur de la fonction J pour ces matrices $\mathbf{S}_1$ et $\mathbf{S}_2$,
- la précision micro-moyennée obtenue en comparant la classification produite à partir de $\mathbf{S}_1$ à la classification réelle des objets.

Si le problème d'optimisation $(\mathcal{P})$ est effectivement pertinent, on devrait alors observer un lien entre la valeur de J et la précision micro-moyennée associée. Plus la valeur de J est faible, plus la précision micro-moyennée devrait être proche de 1. La Fig. 14 représente un nuage de points, doint les abscisses sont les valeurs de J et les ordonnées les précisions micro-moyennées, pour une matrice de relation réelle. La matrice utilisée est celle du premier échantillon du jeu de données documents - mots M2, présenté dans la Sec. IV-1.

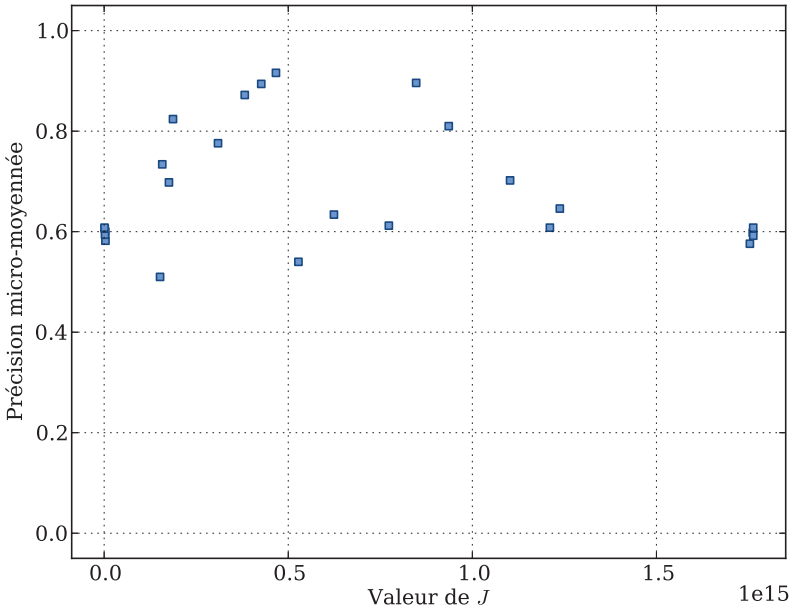


Figure 14 – Précision micro-moyennée en fonction de la valeur de J pour une matrice de relation issue du jeu de données réel M2 présenté dans la Sec. IV-1.

On observe sur la Fig. 14 que la qualité de la classification n'est pas corrélée à la valeur prise par la fonction J, pour des matrices de similarités $\mathbf{S}_1$ et $\mathbf{S}_2$ données. Il semble alors que ce problème d'optimisation ne soit pas directement lié au problème du calcul des similarités pour la tâche de classification automatique, et nous ne poursuivons pas cette étude.

II-5 Conclusion du chapitre

Dans ce chapitre, nous avons présenté l'algorithme χ-Sim qui avait initialement été développé par Bisson et Hussain (2008). Nous avons proposé des améliorations qui concernent principalement la normalisation des valeurs de similarité dans l'intervalle [0,1]. Grâce à cette nouvelle normalisation et en nous appuyant sur les travaux de Qamar et Gaussier (2009) sur la mesure de similarité du Cosinus généralisé, nous avons prouvé que pour certaines valeurs de paramètres de notre algorithme, toutes les matrices de similarité calculées étaient semi-définies positives, proposant ainsi une interprétation algébrique de l'approche.

De plus, nous avons introduit une pseudo-norme k à l'algorithme qui, en conjonction avec l'utilisation d'une étape de seuillage, permet de mieux adapter la méthode à des données de grande dimension, et de réduire la propagation du bruit pendant l'exécution de l'algorithme χ-Sim.

Finalement, nous avons étudié deux pistes pour l'étude et l'amélioration de l'algorithme :

- l'introduction d'un facteur d'amortissement à la méthode, qui permettrait de réduire l'influence négative des chemins redondants, sans succès jusqu'à présent car les chemins redondants semblent avoir une influence positive sur le comportement actuel de l'algorithme χ-Sim ;

- la définition d'un problème d'optimisation sous contraintes sous-jacent à l'algorithme, qui ne semble pas adapté.

Bien que ces deux pistes n'aient pas donné lieu à des améliorations pratiques de l'algorithme χ-Sim pour l'instant, elles ouvrent cependant la voie à de futures études.

Chapitre III

L'algorithme multivue MVSim

Sommaire du chapitre

Dans de nombreuses applications, telles que les réseaux sociaux, les données concernent plus de deux types d'objets, et les méthodes classiques de calcul de similarité et de classification ne sont pas capables de prendre en compte toutes les informations disponibles. En effet, ces méthodes ne peuvent traiter qu'une matrice de données alors que ces données sont le plus souvent constituées d'un ensemble de matrices. Ces données sont dites multivues (ou multirelationnelles) car chaque matrice décrit une relation entre deux types d'objets, et fournit donc une vue sur les données.

Dans ce chapitre, nous présentons alors la seconde contribution de cette thèse, à savoir l'algorithme multivue de calcul de co-similarités MVSim*, pour lequel l'objectif est de calculer M matrices de similarité à partir de N matrices de relations, décrivant les liens entre les instances des M types d'objets X_i présents dans le jeu de données. L'algorithme* MVSim *s'appuie sur l'algorithme de calcul de co-similarité χ-*Sim*, et le généralise au cas des données multivues. Son objectif est alors de prendre en compte les informations provenant des multiples matrices de relations, et de ne pas traiter de façon indépendante ces différentes vues sur les données.*

Après avoir décrit l'algorithme et ses différentes variantes, ainsi que des propriétés sur sa convergence, nous attacherons une importance particulière à l'étude de sa complexité et à sa parallélisation possible. Le fait que cet algorithme soit facilement parallélisable permet d'envisager des applications très intéressantes telles que le traitement de matrices de grandes tailles dans le Chap. IV.

III-1 Description de l'architecture

Nous avons présenté l'algorithme χ-Sim dans le Chap. II, qui travaille à partir d'une matrice de relation unique $\mathbf{R}$ décrivant la relation entre les instances de deux types d'objets X_1 et X_2. Le résultat produit par cet algorithme est deux matrices de co-similarité $\mathbf{S}_1$ et $\mathbf{S}_2$ contenant respectivement les similarités pour les instances de X_1 et de X_2. Dans ce chapitre, les données ne sont plus composées d'une unique matrice de relation, mais de M matrices $\{\mathbf{R}_v\}_{v=1}^{M}$, décrivant les relations entre les N types d'objets $\{X_n\}_{n=1}^{N}$. Finalement, nous souhaitons calculer les co-similarités entre toutes les paires d'instances de ces multiples types d'objets, afin d'éventuellement effectuer une classification d'un ou plusieurs de ces objets. A l'instar de l'algorithme χ-Sim, l'idée est alors que le calcul de chaque matrice de co-similarité soit basé sur les autres matrices de co-similarité.

III-1.1 Vue fonctionnelle de l'algorithme χ-Sim

L'algorithme χ-Sim peut être décrit d'un point de vue fonctionnel comme dans la Fig. 15. Cette figure représente une instance de l'algorithme qui travaille sur la matrice de relation $\mathbf{R}_v$ avec les paramètres de l'algorithme qui sont la pseudo-norme k, le niveau de seuillage p et le nombre d'itérations T. Ces paramètres de χ-Sim seront notés θ dans la suite. Cette instance est alors notée χ-Sim$_v$, et doit être initialisée par deux matrices de similarité $\mathbf{S}_i$ et $\mathbf{S}_j$ – la matrice identité permet d'initialiser l'algorithme quand aucune information *a priori* n'est disponible sur les données – et calcule deux matrices de similarité $\mathbf{S}_i^{[v]}$ et $\mathbf{S}_j^{[v]}$.

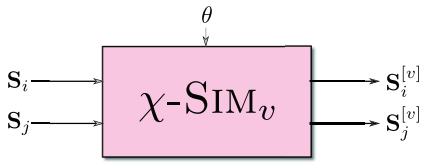


Figure 15 – Vue fonctionnelle d'une instance de l'algorithme χ-Sim.

Comme nous l'avons précisé en introduction, nous considérons dans ce chapitre des données multivues, c'est à dire comportant plusieurs matrices de relation. Nous proposons alors d'associer à chaque matrice de relation $\mathbf{R}_v$ (ou vue V_v) une instance de l'algorithme χ-Sim qui sera notée χ-Sim$_v$. Chacune de ces instances calcule alors deux matrices de similarité pour chacun des types d'objets qu'elle concerne, et nous aurons plusieurs instances pouvant calculer des matrices de similarités d'un même type d'objets. Afin de pouvoir les distinguer, nous noterons $\mathbf{S}_i^{[v]}$ la matrice contenant les similarités calculées entre toutes les instances de X_i à partir de la vue V_v par χ-Sim$_v$. Nous pouvons alors définir l'ensemble des matrices de similarité du type d'objets X_i comme $\mathcal{S}_i = \{\mathbf{S}_i^{[v]} \mid X_i \in V_v\}$ avec $X_i \in V_v$ qui signifie que la vue V_v décrit la relation entre les instances de X_i et d'un autre type d'objets.

Pour illustrer nos propos, reprenons l'exemple de jeu de données multivue « jouet » présenté dans la Fig. 2, qui comporte trois types d'objets X_1, X_2 et X_3 et deux vues V_1 et V_2. A partir de ce jeu de données, nous pouvons donc instancier l'algorithme χ-Sim deux fois, ce qui

donne χ-Sim$_1$ et χ-Sim$_2$, et trois ensembles de matrices de similarité qui seront les suivants : $\mathcal{S}_1 = \{\mathbf{S}_1^{[1]}\}$, $\mathcal{S}_2 = \{\mathbf{S}_2^{[1]}, \mathbf{S}_2^{[2]}\}$ et $\mathcal{S}_3 = \{\mathbf{S}_3^{[2]}\}$.

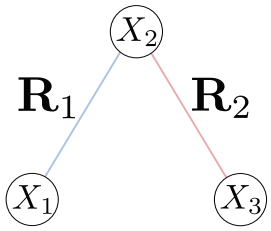


Figure 16 – Exemple d'un jeu de données « jouet », avec 3 types d'objets et 2 relations.

Lorsque l'on travaille avec des données multivues, nous faisons l'hypothèse que ces vues apportent des informations complémentaires pour décrire les objets, qui ne sont pas contradictoires. Plus précisément, cela signifie que toutes les vues ne doivent pas fournir les mêmes similarités, sinon l'intérêt des approches multivues serait nul. De plus, si les vues fournissent des similarités systématiquement contradictoires (si 0 dans une vue, alors 1 dans une autre vue), il ne sera également pas possible de correctement partager l'information entre elles.

L'intérêt de la tâche d'apprentissage multivue de similarités est alors d'avoir un échange d'information entre les vues. Nous choisissons alors d'utiliser le fait que l'algorithme χ-Sim puisse être initialisé par des matrices de similarités. En reprenant notre exemple de la Fig. 2, on peut facilement imaginer que la matrice $\mathbf{S}_2^{[1]}$ serve à initialiser l'entrée correspondante de l'instance χ-Sim$_2$, ou inversement.

III-1.2 Agrégation de matrices de similarité

Pour chaque type d'objet X_i, nous allons disposer de plusieurs matrices de similarités $\mathbf{S}_i$, calculées par les multiples instances de χ-Sim, à partir des différentes vues des données. Il faut alors définir comment agréger les informations provenant de ces différentes matrices. Nous introduisons pour ce faire une fonction générique d'agrégation Agg($\mathcal{S}_i$,θ') qui, à partir d'un ensemble de matrices de similarité d'un même type d'objets X_i – noté $\mathcal{S}_i$ – et de paramètres θ', calcule une matrice de similarité « consensus » notée $\hat{\mathbf{S}}_i$.

Les paramètres de la fonction θ' seront décrits dans la suite, lorsque nous proposerons plusieurs implémentations de la fonction d'agrégation dans la Sec. III-3. La fonction d'agrégation qui sera chargée des matrices de similarité de X_i sera notée Agg$_i$ pour plus de simplicité. La Fig. 17 présente la vue fonctionnelle de cette fonction d'agrégation.

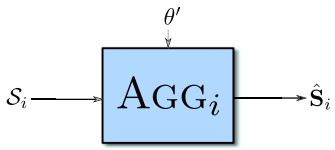


Figure 17 – Vue fonctionnelle de la fonction d'agrégation Agg$_i$.

Ainsi, nous instancions :

- $\forall V \in \mathcal{V}$, telle que $V : X_i \sim X_j$, l'instance χ-Sim$_v$ de l'algorithme de co-similarité χ-Sim, qui calcule les deux matrices de similarités $\mathbf{S}_i^{[v]}$ et $\mathbf{S}_j^{[v]}$.
- $\forall X_i \in \mathcal{X}$, l'instance Agg$_i$ de la fonction générique d'agrégation, qui calcule la matrice de similarité « consensus » $\hat{\mathbf{S}}_i$.

Comme nous l'avons précisé plus tôt, l'intérêt d'utiliser l'algorithme χ-Sim comme notre brique de base pour construire cet algorithme multivue, est qu'il peut être initialisé par une matrice de similarité quelconque. Nous choisissons donc que chaque instance χ-Sim$_v$ (avec $V_v : X_i \sim X_j$), soit initialisée par $\hat{\mathbf{S}}_i$ et $\hat{\mathbf{S}}_j$.

III-1.3 Exemple d'instanciation de l'architecture

La Fig. 18 présente l'instanciation de l'architecture multivue MVSim pour le jeux de données « jouet » de la Fig. 2 décrit précédemment. Dans ce cas très simple, on remarque que les ensembles $\mathcal{S}_1$ et $\mathcal{S}_3$ se réduisent à une seule matrice de similarité, car X_1 et X_3 ne sont chacun impliqué que dans une seule vue. Nous supposons que même dans ce cas, nous utilisons une instance de la fonction Agg, et nous verrons après avoir défini cette fonction, que cela est nécessaire pour MVSim. L'échange d'information entre les vues se fera alors par le biais de la matrice de similarité « consensus » de X_2 (impliquée dans les deux vues) notée $\hat{\mathbf{S}}_2$.

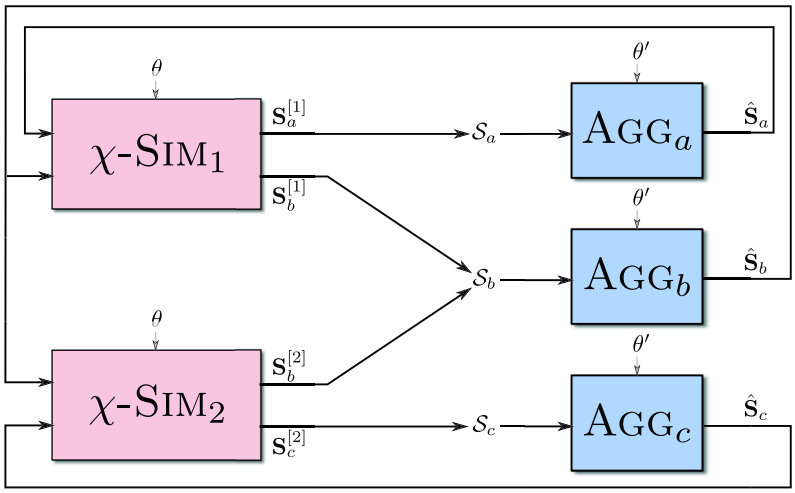


Figure 18 – Instanciation de l'architecture pour le jeux de données décrit par la Fig. 2

Maintenant que nous avons présenté l'idée de base de l'algorithme MVSim, ainsi qu'une vue fonctionnelle de celui-ci, nous allons développer dans les sections suivantes :

- Quelle dynamique utiliser pour MVSim ? Dans quel ordre sont exécutées les différentes fonctions ?
- Comment définir la fonction d'agrégation Agg, et ses paramètres θ' ?
- Comment choisir les paramètres θ des instances de χ-Sim ?

III-2 Dynamique de l'architecture

Nous avons défini comment instancier l'architecture de l'algorithme MVSim, nous allons maintenant examiner plusieurs politiques pour l'ordre dans lequel les fonctions et algorithmes de MVSim seront exécutés. Nous considérons deux approches : asynchrone et synchrone, détaillées dans la suite.

Politique asynchrone

La politique asynchrone consiste à utiliser un ordre, pré-établi ou non, pour exécuter successivement les fonctions de MVSim. Ainsi, on pourra par exemple décider d'exécuter χ-Sim$_v$ puis d'utiliser l'une des matrices de similarité calculées pour initialiser une autre instance χ-Sim$_w$.

L'inconvénient de cette approche est qu'il n'y a pas d'équité dans l'exploitation des différentes vues, et donc certaines d'entre elles seront peut-être privilégiées. Dans des travaux préliminaires (Grimal et Bisson 2010), nous avons utilisé une « cascade » de deux instances de χ-Sim réalisant chacune 4 itérations sur leur matrice de relation respective, sans fonction d'agrégation, et avons observé que l'ordre choisi pour les deux instances joue un rôle très important dans les résultats obtenus. En effet, la seconde instance était significativement privilégiée par rapport à la première, en cela que les résultats obtenus avec la cascade étaient proches de ceux obtenus en utilisant uniquement la seconde instance.

Nous nous intéressons dans cette thèse à des tâches d'apprentissage non supervisé de similarités, et nous ne disposons d'aucune connaissance *a priori*, il est donc impossible de décider si une vue fournit des informations plus pertinentes qu'une autre dans notre cas. Cependant, dans une situation où l'on dispose de connaissances *a priori* sur la pertinence des informations des vues pour calculer les similarités des objets, cette approche peut se révéler intéressante. On pourra alors chercher à privilégier les vues fournissant les informations les plus pertinentes.

Politique synchrone

Contrairement à la politique asynchrone, l'approche synchrone suppose que toutes les vues sont d'égale pertinence, et doivent être traitées de façon symétrique, sans distinction. C'est donc la politique que nous choisirons pour nos expérimentations dans le Chap. IV. Avec cette politique, l'algorithme MVSim exécute alternativement toutes les instances de χ-Sim pour calculer les ensembles de matrices de similarités, puis toutes les instances de la fonction Agg afin d'obtenir une matrice de similarité « consensus » $\hat{\mathbf{S}}_i$ pour chaque type d'objets X_i.

Comme dans le Chap. II, nous ne disposons d'aucune connaissance *a priori* sur les similarités des objets, nous initialisons par conséquent les matrices de similarité par des matrices identité (chaque objet est similaire à lui-même, et il n'est similaire à aucun autre). Par la suite, les matrices de similarité « consensus » seront utilisées pour ré-initialiser les instances χ-Sim$_v$. Nous pouvons alors définir une itération de l'algorithme MVSim comme l'exécution de toutes les instances des χ-Sim$_v$, puis l'exécution de toutes les fonctions Agg$_i$, et nous notons alors

T_{MV} le nombre maximal d'itérations (afin de ne pas le confondre avec le nombre d'itérations T de l'algorithme χ-Sim).

Précisons finalement qu'un critère d'arrêt utilisant le nombre maximal d'itérations, ainsi qu'un paramètre de convergence des matrices de similarités pourra être défini une fois que nous aurons précisé la fonction d'agrégation que nous allons utiliser.

III-3 Fonction d'agrégation

Dans cette section nous définissons plus précisément la fonction d'agrégation Agg introduite précédemment. Le rôle de cette fonction d'agrégation est de fusionner différentes matrices de similarité, qui ont été produites par plusieurs instances de l'algorithme χ-Sim, à partir des multiples vues du jeu de données. Nous parlons alors de matrice de similarité « consensus » car la matrice que cette fonction calcule doit représenter l'ensemble des informations sur les similarités des objets extraites des différentes vues.

Dans un cadre non supervisé et sans connaissance *a priori* ni sur les objets, ni sur les différentes vues, on peut considérer deux familles de stratégies pour l'agrégation des similarités :

- des opérations terme-à-terme de matrice, telles que le maximum, le minimum et la moyenne ;
- des méthodes basées sur les fusions de classements.

Les méthodes basées sur les fusions de classements consistent à trier les similarités dans un premier temps, afin d'obtenir autant de classements, que de matrices de similarités. Puis ces méthodes cherchent à obtenir un classement « consensus », qui désordonne le moins possible les rangs initiaux des similarités. On peut finalement reconstruire une matrice de similarité à partir de ce classement « consensus », en répartissant les valeurs des similarités en fonction de leur rang dans ce classement.

Nous allons alors définir une forme générique pour cette fonction d'agrégation, et voir dans la suite quelles valeurs nous utiliserons pour nos expérimentations. Rappelons donc que nous notons la fonction d'agrégation pour le type d'objets X_i comme $\text{Agg}(\mathcal{S}_i,\theta')$. De plus, nous avons vu dans la section sur la dynamique de MVSim que la matrice de similarité « consensus » $\hat{\mathbf{S}}_i$ sera calculée à chaque itération de l'algorithme, nous ajoutons donc l'exposant (t) pour l'itération t. La fonction d'agrégation est alors définie comme suit :

$$\begin{aligned} \text{Agg}(\mathcal{S}_i^{(t)},\theta') &= \hat{\mathbf{S}}_i^{(t)} \\ &= (1-\alpha)\,\hat{\mathbf{S}}_i^{(t-1)} + \alpha\,\text{F}(\mathcal{S}_i^{(t)}) \end{aligned} \tag{19}$$

avec $\alpha \in [0,1]$ un paramètre de θ', et $\text{F}(\mathcal{S})$ une opération d'agrégation élémentaire sur les matrices de l'ensemble $\mathcal{S}$ avec la seule contrainte que la matrice calculée ait ses valeurs comprises entre 0 et 1. La fonction d'agrégation est alors la moyenne pondérée entre la matrice de similarité « consensus » de l'itération précédente $\hat{\mathbf{S}}_i^{(t-1)}$ et l'agrégation élémentaire des matrices de similarité courante $\text{F}(\mathcal{S}_i^{(t)})$.

Finalement, on peut préciser que l'Eq. (19) reste volontairement très générique et permet de couvrir de nombreuses variantes, qui seront fonction du paramètre α et de la fonction d'agrégation élémentaire F.

III-4 Paramètres et convergence

Jusqu'à présent, nous avons introduit les paramètres suivants pour l'algorithme MVSim :

- pour chaque instance de χ-Sim : θ qui comprend le nombre d'itérations T, le paramètre de pseudo-norme k, le seuillage p ;
- pour chaque fonction d'agrégation Agg : θ' qui comprend le paramètre de moyenne α et la fonction d'agrégation élémentaire F ;
- pour l'algorithme MVSim lui-même : le nombre d'itération T_{MV}, et éventuellement un critère d'arrêt.

Afin de réduire le nombre de paramètres, nous choisissons d'imposer que les paramètres θ de chaque instance de χ-Sim soient les mêmes, et nous prendrons toujours $T = 1$, les itérations se faisant maintenant au niveau de MVSim.

A l'instar des paramètres θ de l'algorithme χ-Sim, nous imposons que les paramètres θ' de toutes les fonctions d'agrégation soient les mêmes. Nous allons détailler dans la Sec. III-4.1 les différentes fonctions d'agrégation élémentaire qui seront utilisées dans la partie expérimentale, puis dans la Sec. III-4.2 nous préciserons quelles valeurs nous avons choisies pour le paramètre α afin de garantir la convergence de l'algorithme.

III-4.1 Fonction d'agrégation élémentaire F

Les différentes fonctions d'agrégation élémentaire qui seront utilisées dans la partie expérimentale sont :

- le minimum terme-à-terme,
- le maximum terme-à-terme,
- la moyenne terme-à-terme,
- la fonction CSPA dérivée de la méthode de consensus de classification présentée dans la Sec. I-4.2.

Les trois premières fonctions sont clairement des opérations terme-à-terme sur les matrices de similarités, tandis que la dernière fonction est basée sur une méthode de fusion de classement. Le pseudo-code de la fonction élémentaire d'agrégation est donné dans l'Alg. 3. Le principe de cette dernière fonction d'agrégation est d'utiliser un algorithme de classification (dans notre cas, un algorithme de CAH) sur chacune des matrices $\mathbf{S}_n^{[v]}$, puis de construire la matrice de similarité « consensus » en comptant le nombre de fois où deux objets sont dans la même classe.

Chacune des fonctions élémentaires basées sur des opérations terme-à-terme est associée à une sémantique différente. Si l'on choisit d'utiliser le maximum par exemple, alors deux objets seront similaires s'ils sont similaires dans au moins une des vues du jeu de données ; alors que le minimum implique que deux objets sont similaires s'ils sont similaires dans toutes les vues.

Algorithme 3 La fonction d'agrégation élémentaire basée sur CSPA.

Entrées : Un ensemble $\mathcal{S}_n = \{\mathbf{S}_n^{[v]}\}_{v=1}^{H}$ et un nombre de classes K
Sorties : La matrice de similarité « consensus » $\hat{\mathbf{S}}_n$ de terme générale $\hat{s}_{ij}$
  $\hat{\mathbf{S}}_n \leftarrow \mathbf{I}_{\eta_n}$ *(initialisation)*
  Pour $v = 1 \rightarrow H$ **:**
    $\pi_v \leftarrow \text{CAH}(\mathbf{S}_n^{[v]}, K)$ *(classification ascendante hiérarchique)*
    Pour $i,j : 1 \rightarrow \eta_n$ **:**
      Si $\pi_v(i) = \pi_v(j)$ **alors**
        $\hat{s}_{ij} \leftarrow \hat{s}_{ij} + 1/H$ *($x_i^{(n)}$ et $x_j^{(n)}$ sont dans la même classe pour V_v)*
      Fin
    Fin
  Fin

Nous avons présenté le principe des méthodes basées sur la fusion de classements dans la Sec. III-3, qui se concentrent sur le classement des valeurs de similarités, et construisent un classement « consensus » qui cherche à préserver le plus possible les classements initiaux. Cependant, notons que ces approches ignorent les valeurs numériques des similarités pour s'intéresser uniquement à l'ordre entre ces valeurs. Ce choix ne paraît pas nécessairement judicieux dans notre contexte, et nous comparerons donc les performances de la fonction CSPA, à celles des autres fonctions élémentaires d'agrégation pour confirmer cette intuition.

> ***Remarque 6.*** *Nous avons vu dans le Chap. II que pour $k = 1$ et $p = 0$, les matrices de similarités calculées par l'algorithme χ-Sim sont toutes semi-définies positives (SDP). Remarquons ici que l'algorithme* MVSim *préservera cette propriété si la fonction élémentaire d'agrégation utilisée est la moyenne, car la multiplication d'une matrice SDP par un scalaire est SDP, et que la somme de deux matrices SDP est SDP également. Il n'est en revanche pas possible de fournir de telle garantie pour le minimum, le maximum ou la fonction CSPA.*

III-4.2 Paramètre α et convergence de MVSim par amortissement

La paramètre α chargé de contrôler la pondération de la moyenne entre la matrice de similarité « consensus » de l'itération précédente et le résultat de l'agrégation élémentaire des similarités courantes, joue un rôle très important dans le comportement de l'algorithme MVSim. Remarquons premièrement que pour $\alpha = 0$, les similarités courantes sont ignorées, ce qui implique que les matrices de similarités « consensus » $\hat{\mathbf{S}}_i$ n'évoluent plus. Inversement, pour $\alpha = 1$, c'est la matrice de similarité « consensus » de l'itération précédente qui sera ignorée, et la fonction Agg sera alors égale à la fonction d'agrégation élémentaire F.

Afin d'obtenir un comportement plus intéressant de l'algorithme MVSim, il est judicieux de faire en sorte que ce paramètre α soit dynamique, i.e. qu'il dépend de l'itération courante t, on le notera donc $\alpha(t)$. En effet, nous avons décrit le comportement de la fonction d'agrégation pour des valeurs extrêmes de α, et si nous souhaitons que MVSim converge, nous pouvons faire décroître α avec t. La façon la plus naturelle d'obtenir ce comportement est de définir $\lambda \in [0,1[$ un paramètre d'amortissement, et de fixer $\alpha(t) = \lambda^{t-1}$. Ainsi, pendant la première itération ($t = 1$), la matrice consensus $\hat{\mathbf{S}}_i^{(1)}$ sera simplement égale au résultat de la fonction élémentaire

d'agrégation $\mathrm{F}(\mathcal{S}_i^{(1)})$. Puis au cours des itérations de l'algorithme, le poids va progressivement glisser vers la matrice de similarité « consensus » précédente, impliquant la convergence de l'algorithme. La Fig. 19 illustre l'évolution pendant 16 itérations de l'algorithme, des poids $\alpha(t)$ et $1-\alpha(t)$ pour $\lambda = 0.8$. L'Eq. (19) générique devient alors :

$$\hat{\mathbf{S}}_i^{(t)} = (1-\lambda^{t-1})\,\hat{\mathbf{S}}_i^{(t-1)} + \lambda^{t-1}\,\mathrm{F}(\mathcal{S}_i^{(t)}) \tag{20}$$

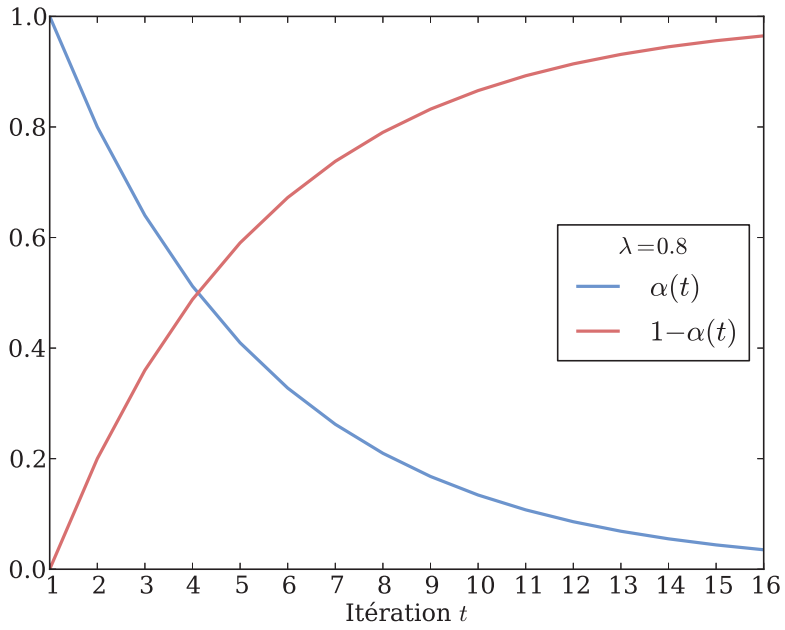


Figure 19 – Évolution des poids $\alpha(t)$ et $1-\alpha(t)$ en fonction de t, pour $\alpha(t) = \lambda^{t-1}$.

Un autre schéma pour la fonction d'agrégation a été envisagé, qui correspond mieux à un amortissement des chemins d'ordres supérieurs, en fonction de leur longueur (voir la Sec II-3.2 sur les chemins redondants et l'amortissement pour χ-Sim) et qui donne, sans normalisation :

$$\begin{aligned}\hat{\mathbf{S}}_i^{(t)} &= \sum_{t'=1}^{t} \lambda^{t'-1}\,\mathrm{F}(\mathcal{S}_i^{(t')}) \\ &= \hat{\mathbf{S}}_i^{(t-1)} + \lambda^{t-1}\,\mathrm{F}(\mathcal{S}_i^{(t)})\end{aligned}$$

S'il l'on veut normaliser les valeurs de similarités entre 0 et 1, on peut utiliser la formule suivante :

$$\hat{\mathbf{S}}_i^{(t)} = \frac{\hat{\mathbf{S}}_i^{(t-1)} + \lambda^{t-1}\,\mathrm{F}(\mathcal{S}_i^{(t)})}{1+\lambda^{t-1}} \tag{21}$$

ce qui correspond, selon l'équation générique pour la fonction d'agrégation (19), à prendre $\alpha(t) = \frac{\lambda^{t-1}}{1+\lambda^{t-1}}$. Avec ce schéma là, la convergence peut également être garantie si λ appartient à l'intervalle $[0,1[$, car la fonction F est bornée et le facteur d'amortissement λ^{t-1} est exponentiellement décroissant.

Finalement, nous utiliserons comme fonction d'agrégation soit l'Eq. (20) soit l'Eq. (21) dans la partie expérimentale au Chap. IV, et le nombre d'itérations T_{MV} de l'algorithme MVSim sera généralement fixé entre 10 et 20. Quant au critère d'arrêt, il consiste à comparer le maximum des normes de Frobenius des différences entre les matrices de similarité « consensus » de l'itération courante et précédente, par rapport à un paramètre ϵ fixé. Plus formellement, nous nommons cette quantité $d^{(t)}$, et nous la calculons comme suit :

$$d^{(t)} = \max_{1 \leqslant i \leqslant N} \|\hat{\mathbf{S}}_i^{(t)} - \hat{\mathbf{S}}_i^{(t-1)}\|_2 \tag{22}$$

Ainsi, si toutes les matrices de similarité « consensus » sont stables, l'algorithme peut s'arrêter.

III-5 Algorithme

Nous donnons dans l'algorithme 4 le pseudo-code pour l'algorithme MVSim. L'étape de seuillage des $p\%$ des similarités les plus faibles n'est dorénavant plus réalisée par les instances de χ-Sim, mais après l'étape d'agrégation. En effet, la similarité entre deux objets peut être faible dans une vue, mais si cela n'est pas le cas dans les autres vues, l'algorithme sous-estimerait cette valeur. Ainsi, c'est seulement si une similarité est faible dans la matrice de similarité « consensus » qu'elle sera susceptible d'être mise à zéro.

Par ailleurs, on peut considérer que l'algorithme MVSim est une généralisation multivue de l'algorithme χ-Sim. En effet, considérons que $M = 1$ et que $N = 2$, afin de n'avoir qu'une seule vue décrivant la relation entre deux types d'objets. Il suffit de fixer $T_{\text{MV}} = T$, $\alpha(t) = 1$, et de considérer que la fonction élémentaire d'agrégation F d'une seule matrice, renvoie simplement cette matrice tel quel. Dans ce cas alors, on retrouve exactement l'algorithme χ-Sim. Pour aller plus loin, on peut reprendre les conditions précédentes auxquelles on retire la condition sur $\alpha(t)$, et on retrouve l'idée développée dans la Sec. II-3.2 pour introduire de l'amortissement dans l'algorithme χ-Sim.

III-6 Complexité et parallélisation

Dans cette section, nous allons étudier la complexité de l'algorithme MVSim et présenter comment il est possible d'exploiter les architectures matérielles à multiples nœuds de calcul qui deviennent de plus en plus courantes, afin de répartir les calculs et en diminuer la complexité en temps et en espace. Par nœud de calcul, nous entendons soit un processeur, soit le cœur d'un processeur. De plus, nous nous intéresserons à l'application de MVSim au traitement de matrices de grandes dimensions, en exploitant cette propriété de parallélisation.

III-6.1 Complexité de l'algorithme MVSim

La complexité de l'algorithme MVSim est évidemment liée à la complexité des briques de base dont il est composé, à savoir l'algorithme χ-Sim, et la fonction d'agrégation Agg. Dans un premier temps, considérons que les M matrices de relations sont toutes de taille $\eta \times \eta$. Nous avons déjà vu dans dans la Sec. II-2.3, que la complexité temporelle de χ-Sim est dans ce cas en $\mathcal{O}(\eta^3)$, et sa complexité spatiale est en $\mathcal{O}(\eta^2)$. La fonction d'agrégation a quant à

Algorithme 4 L'algorithme multivue de calcul de co-similarités MVSim

Entrées : Les matrices de relation $\{\mathbf{R}_v\}_{v=1}^{M}$, paramètres : T_{MV}, F, $\alpha(t)$, k, p

Sorties : Les matrices de similarités $\{\mathbf{S}_n\}_{n=1}^{N}$

(Initialisation)

$\epsilon \leftarrow 10^{-2}$

$t \leftarrow 0$

Pour $n = 1 \rightarrow N$ **:**

$\hat{\mathbf{S}}_n^{(t)} \leftarrow \mathbf{I}_{\eta_n}$

Fin

(Itérations de l'algorithme)

Répéter :

$t \leftarrow t + 1$

Pour $v = 1 \rightarrow M$ **:**

(Ré-)initialiser χ-Sim$_v$ avec $\hat{\mathbf{S}}_i^{(t-1)}$ et $\hat{\mathbf{S}}_j^{(t-1)}$

Calcul de $\mathbf{S}_i^{[v]}$ et $\mathbf{S}_j^{[v]}$ par χ-Sim$_v(T = 1, k)$

Fin

Pour $n = 1 \rightarrow N$ **:**

$\hat{\mathbf{S}}_n^{(t)} \leftarrow \mathrm{Agg}(\mathcal{S}_n^{(t)}, \mathrm{F}, \alpha(t))$ selon Eq. (20) ou Eq. (21)

Seuillage des $p\%$ similarités les plus faibles de $\hat{\mathbf{S}}_n^{(t)}$

Fin

$d^{(t)} \leftarrow \max_{1 \leqslant n \leqslant N} \|\hat{\mathbf{S}}_n^{(t)} - \hat{\mathbf{S}}_n^{(t-1)}\|_2$

Jusqu'à $t \geqslant T_{\mathrm{MV}}$ **et** $d^{(t)} < \epsilon$

(Affectation finale des sorties)

Pour $n = 1 \rightarrow N$ **:**

$\mathbf{S}_n \leftarrow \hat{\mathbf{S}}_n^{(t)}$

Fin

elle une complexité temporelle et spatiale en $\mathcal{O}(\eta^2)$, peu importe la fonction F choisie parmi celles proposées précédemment.

Ainsi, pour M matrices de relations et N types d'objets différents, la complexité temporelle de l'algorithme MVSim est en $\mathcal{O}(M\,\eta^3 + N\,\eta^2) = \mathcal{O}(M\,\eta^3)$. Sa complexité spatiale est associée au fait que l'algorithme doit garder en mémoire toutes les matrices de similarités (qui ne sont pas nécessairement creuses [6]), elle est donc en $\mathcal{O}(2\,M\,\eta^2)$.

L'algorithme est cependant facilement parallélisable, il suffit pour cela d'affecter à chaque vue, donc à chaque instance de χ-Sim, un nœud de calcul. En assignant ainsi à chaque instance de l'algorithme χ-Sim de MVSim un nœud de calcul, la complexité temporelle de l'approche devient $\mathcal{O}(\eta^3)$, la même que pour l'algorithme χ-Sim. On néglige évidemment dans cette étude les temps de transferts de données qui devront être effectués à chaque itération de l'algorithme, afin de procéder à l'agrégation des matrices de similarités.

Nous allons maintenant voir dans la section suivante, comment tirer profit de cette parallélisation simple de l'algorithme MVSim.

III-6.2 Application au traitement d'une matrice de grande taille

La motivation pour le développement de l'algorithme MVSim est de pouvoir traiter des données multivues, afin de combiner les informations provenant des différentes vues d'un problème. Cependant, nous proposons dans cette section de l'utiliser pour traiter une matrice de grande taille, pour laquelle l'utilisation directe de l'algorithme χ-Sim ne serait pas possible, soit à cause du temps de calcul trop long, soit à cause du stockage de matrices de similarités trop grandes pour être mémorisées dans la mémoire vive.

Nous proposons donc de découper une matrice de relation de grande taille, en plusieurs blocs, qui seront assimilés à de multiples vues par l'algorithme MVSim. Suivant les choix effectués pour le découpage, ainsi que le type d'objets que l'on cherche à classifier automatiquement, on verra alors différentes variantes.

III-6.2.a Découpage selon une seule dimension

Dans un premier temps, supposons que l'on dispose d'une matrice de relation $\mathbf{R}$, de taille $\eta_1 \times \eta_2$, entre des documents et des mots (l'approche proposée reste bien entendu valide pour d'autres types d'objets). De plus on fait l'hypothèse que l'on cherche uniquement à classifier les documents et que le nombre de mots est très important : $\eta_2 \gg \eta_1$. Dans ce cas si η_1 et η_2 sont relativement grands, le temps de traitement par l'algorithme χ-Sim sera très long, et le stockage de la matrice de similarité des η_2 mots $\mathbf{S}_2$ posera problème.

6. Dans le cas général, les matrices de similarité calculées par les instances de χ-Sim ne seront plus creuses après 2 ou 3 itérations. Cela peut tout de même être le cas avec un paramètre de seuillage élevé ou la fonction élémentaire d'agrégation du minimum.

Méthode

C'est pourquoi, alors que l'on cherche uniquement à classifier les documents, nous proposons de ne pas calculer cette matrice $\mathbf{S}_2$ intégralement, afin d'accélérer le calcul de $\mathbf{S}_1$ et d'économiser une grande quantité de mémoire vive. Nous présentons alors l'approche suivante, dans laquelle nous découpons l'ensemble des η_2 mots en M sous-ensembles de même taille η_2/M. Nous obtenons ainsi M matrices de relations $\mathbf{R}_v$ de taille $\eta_1 \times \eta_2/M$, et nous utilisons ensuite l'algorithme MVSim à partir de ces matrices pour calculer la matrice de similarité entre les documents $\mathbf{S}_1$. Dans ce cas on considère que $N = M + 1$, car chaque sous-ensemble de mots est considéré comme un type d'objets. La Fig. 20 présente une illustration de cette approche.

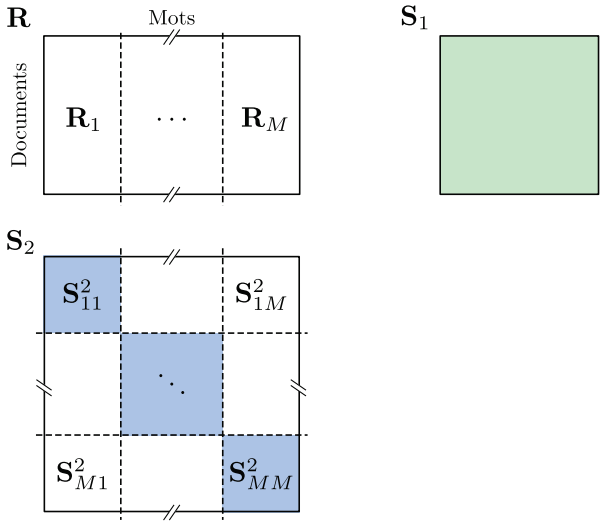


Figure 20 – Illustration du découpage des colonnes d'une matrice en M blocs. Seuls les blocs diagonaux de la matrice de similarité des mots $\mathbf{S}_2$, notés $\mathbf{S}^2_{ii}$, sont alors calculés. La matrice de similarité des documents $\mathbf{S}_1$ est quant à elle entièrement connue.

Etant donné que nous ne découpons pas l'ensemble des documents, la matrice de similarité des documents $\mathbf{S}_1$ est intégralement calculée par l'algorithme MVSim, et permet donc de procéder à une classification de ceux-ci avec n'importe quel algorithme classique.

Complexité temporelle et spatiale

Maintenant que nous avons explicité comment utiliser MVSim dans ce cas, voyons quel est le gain en terme de complexité, à la fois temporelle et spatiale. Si l'algorithme χ-Sim est utilisé sur cette matrice $\mathbf{R}$ de taille $\eta_1 \times \eta_2$ (rappelons que $\eta_2 \gg \eta_1$), la complexité temporelle serait en $\mathcal{O}(\eta_2^3)$ et la complexité spatiale en $\mathcal{O}(\eta_2^2)$. Après découpage de la matrice $\mathbf{R}$, rappelons que les blocs sont de taille $\eta_1 \times \eta_2/M$. Or $\max(\eta_1, \frac{\eta_2}{M}) \leqslant \frac{\eta_2}{M}$, on peut donc dire que la complexité temporelle de cette approche est – au pire si $\frac{\eta_2}{M} > \eta_1$ – en $\mathcal{O}(\frac{\eta_2^3}{M^3})$, elle est donc divisée par un facteur de M^3. Si jamais $\frac{\eta_2}{M} < \eta_1$, cette complexité est alors en $\mathcal{O}(\eta_1^3)$. Quant à la complexité spatiale, bien que les matrices de similarité entre mots soient M^2 fois plus petites, chaque vue calculant sa propre matrice $\mathbf{S}_1$, elle est réduite à $\mathcal{O}(\frac{\eta_2^2}{M})$, elle est donc divisée par un facteur de M.

> ***Remarque 7.*** *Si l'on ne dispose pas de M nœuds de calcul, et que l'on ne parallélise donc pas l'exécution de l'algorithme* MVSim*, la complexité temporelle est tout de même divisée par un facteur M^2, et la complexité spatiale toujours par un facteur M, par le simple fait que l'on travaille sur des matrices de plus petites tailles.*

Conclusion relative au découpage selon une dimension

Classiquement, dans les problèmes de classification automatique de documents, une étape de sélection des mots est effectuée pendant le pré-traitement, afin de ne pas utiliser de mots trop peu informatifs, mais également afin de réduire le temps de calcul et la mémoire nécessaire. Grâce à notre proposition d'utiliser l'algorithme multivue de calcul de co-similarité MVSim, il est possible de garder un plus grand nombre de mots si cela est utile, sans pour autant sacrifier le temps et la mémoire nécessaire à la classification. Evidemment, le fait de découper la matrice de départ en plusieurs blocs implique que les similarités calculées prendront en compte moins d'informations que sans découpage, et induiront donc certainement une qualité de classification moindre. Nous verrons dans le Chap. IV qu'il y a en effet un compromis à trouver entre rapidité du calcul et qualité de classification, mais que cette approche peut permettre d'obtenir des résultats intéressants.

III-6.2.b Découpage selon les deux dimensions

Dans la section précédente, nous avons vu comment découper les colonnes d'une matrice $\mathbf{R}$ dont le nombre de colonnes serait beaucoup plus important que le nombre de lignes. Nous allons maintenant étendre ce problème au cas où l'on veut travailler sur une matrice de grande taille en la découpant à la fois en lignes et en colonnes. Par analogie avec ce qui se passe avec les similarités entre mots dans le cas du découpage des colonnes seulement, le problème qui va se poser est que l'on ne disposera pas de l'intégralité de la matrice de similarité entre documents $\mathbf{S}_1$, mais uniquement de ses blocs diagonaux. Nous allons proposer une méthode capable dans ce contexte de calculer la similarité entre n'importe quelle paire de documents et mesurer le gain en terme de complexité que cette approche apporte.

Méthode

Supposons que l'on dispose d'une matrice de relation $\mathbf{R}$ carrée de taille $\eta \times \eta$ entre des documents et des mots, et que l'on cherche à classifier les documents. Nous la supposons carrée pour simplifier les notations et la description de l'approche, mais il est bien sûr possible de généraliser la méthode à une matrice de taille quelconque. On suppose qu'il n'est pas possible d'utiliser l'algorithme χ-Sim directement sur cette matrice car η est trop grand et cela entraînerait à la fois un trop grand temps de calcul et surtout une occupation mémoire rédhibitoire (deux matrices de similarités de taille η^2).

Dans cette approche, nous proposons donc de découper à la fois l'ensemble des η mots en M sous-ensembles de taille η/M, et l'ensemble des η documents en M sous-ensembles de taille η/M également[7]. Nous obtenons un ensemble de M^2 matrices de relations que nous

7. Nous choisissons de découper les lignes et les colonnes en M blocs afin de simplifier les notations, mais l'approche se généralise si les colonnes sont découpées en M blocs et les lignes en M' blocs.

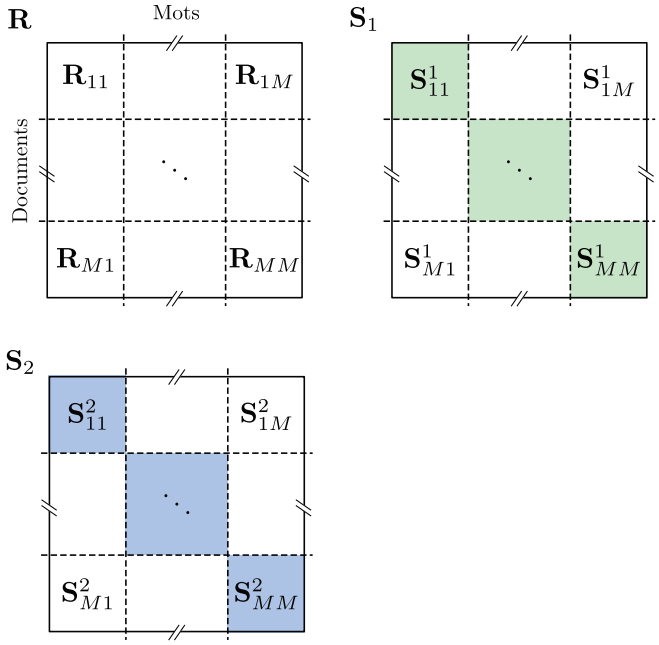


FIGURE 21 – Illustration du découpage des lignes et des colonnes d'une matrice en M^2 blocs. Seuls les blocs diagonaux colorés des matrices de similarités sont calculés.

notons $\{\mathbf{R}_{ij}\}_{i,j=1}^{M}$, toutes de taille $\frac{\eta}{M} \times \frac{\eta}{M}$. Nous utilisons alors l'algorithme MVSim à partir de ces matrices pour calculer les blocs diagonaux des matrices de similarité $\mathbf{S}_1$ et $\mathbf{S}_2$, qui seront respectivement notés $\mathbf{S}^1_{ii}$ et $\mathbf{S}^2_{ii}$. Dans ce cas, on considère que $N = 2\,M$, car chaque sous-ensemble de mots et chaque sous-ensemble de documents est considéré comme un type d'objets. La Fig. 21 présente une illustration de cette approche.

Comme nous l'avons déjà souligné, nous ne disposons pas de la matrice de similarité $\mathbf{S}_1$ complète, mais simplement de ses blocs diagonaux. En effet, nous avons découpé l'ensemble des η documents en M sous-ensembles, et ne calculons avec MVSim que les similarités entre les documents d'un même sous-ensemble. Ainsi, si l'on souhaite connaître la mesure de similarité entre deux documents $x_a^{(1)}$ et $x_b^{(1)}$, deux cas de figures peuvent se présenter :

- $x_a^{(1)}$ et $x_b^{(1)}$ sont dans le même sous-ensemble h de documents (ou bloc), et leur similarité est simplement accessible dans $\mathbf{S}^1_{hh}$;
- $x_a^{(1)}$ et $x_b^{(1)}$ ne sont pas dans le même sous-ensemble de documents, et on doit calculer leur similarité.

Pour calculer cette similarité s^1_{ab}, nous proposons simplement d'utiliser l'Eq. (15) définissant la pseudo similarité du Cosinus généralisé, à partir de $\mathbf{S}_2$, la matrice partielle de similarité entre mots :

$$s^1_{ab} = \frac{\left\langle \mathbf{r}_{a:}, \mathbf{r}_{b:} \right\rangle^k_{\mathbf{S}_2}}{\sqrt{\|\mathbf{r}_{a:}\|^k_{\mathbf{S}_2} \times \|\mathbf{r}_{b:}\|^k_{\mathbf{S}_2}}} \tag{23}$$

Cette similarité doit bien sûr être considérée comme une approximation de la mesure que l'on obtiendrait en utilisant directement l'algorithme χ-Sim. Cette approximation sera de plus dépendante de M, et sera d'autant plus proche de cette valeur que M sera faible. En pratique,

on peut calculer n'importe quel bloc de $\mathbf{S}_1$, noté $\mathbf{S}^1_{ij}$ en commençant par calculer :

$$\begin{aligned}\overline{\mathbf{S}}^1_{ij} &= \mathbf{R}_{i:}\,\mathbf{S}^2\,\mathbf{R}^\top_{j:} \\ &= \sum_{c=1}^{M} \mathbf{R}_{ic}\,\mathbf{S}^2_{cc}\,\mathbf{R}^\top_{jc}\end{aligned}$$

avec $\mathbf{R}_{i:}$ la matrice constituée des lignes du bloc i de lignes et de toutes les colonnes de $\mathbf{R}$. Il faut ensuite normaliser ces valeurs pour qu'elles appartiennent à l'intervalle [0,1], en utilisant les termes diagonaux de $\overline{\mathbf{S}}^1_{ii}$.

Complexité temporelle

Maintenant que nous avons décrit notre proposition, étudions le gain qu'elle offre en terme de complexité temporelle. Le découpage de $\mathbf{R}$ nécessite de tirer aléatoirement η/M éléments, M fois, une fois pour les lignes et une fois pour les colonnes. Puis, il suffit pour chacun de ses M^2 blocs, de créer la sous-matrice correspondante. La complexité temporelle de cette étape est donc de $2\,\eta + M^2 * (\eta/M)^2 = \mathcal{O}(\eta^2)$.

Rappelons ici que si l'algorithme χ-Sim est utilisé sur cette matrice $\mathbf{R}$ de taille $\eta \times \eta$, la complexité temporelle est en $\mathcal{O}(\eta^3)$. Après découpage de la matrice $\mathbf{R}$, la complexité temporelle d'une itération de MVSim sur ces M^2 matrices de taille $\eta/M \times \eta/M$ est alors en $\mathcal{O}(\frac{\eta^3}{M^3})$, elle est donc divisée par un facteur M^3. Précisons que pour que ce résultat soit valable, il faut disposer de M^2 nœuds de calculs pour exécuter en parallèle les M^2 instances de χ-Sim, et aussi les $2\,M$ instances de Agg. Cependant, même si l'on ne parallélise pas l'exécution de MVSim, la complexité temporelle sera tout de même divisée par un facteur M.

Finalement, il faut aussi prendre en compte la complexité du calcul des similarités des paires de documents n'étant pas dans le même sous-ensemble. Ce calcul, s'il est effectué par blocs, a une complexité en $\mathcal{O}(M^3\,\eta^3)$. La complexité temporelle de l'ensemble de l'approche reste en $\mathcal{O}(\eta^3)$, et on ne peut donc pas réduire le temps de calcul en augmentant le nombre de blocs. Néanmoins, si l'on ne désire pas nécessairement connaître la similarité entre toutes les paires de documents, mais seulement ponctuellement, pour une paire donnée, alors on peut la calculer grâce à l'Eq. (23), et le temps d'exécution de MVSim peut lui être réduit en augmentant M.

Complexité spatiale

Considérons maintenant la complexité spatiale de la méthode proposée, en se souvenant que si l'algorithme χ-Sim est directement utilisé sur la matrice $\mathbf{R}$, il faudrait alors mémoriser deux matrices de similarité de taille $\eta \times \eta$, qui ne sont pas nécessairement creuses, ce qui pose problème dès que η devient grand.

Pendant l'exécution de MVSim, chacun des M^2 nœuds de calcul qui exécute une instance de l'algorithme χ-Sim doit avoir en mémoire son bloc de la matrice de données, et les deux matrices de similarité, et une matrice intermédiaire de même taille, ce qui donne une complexité

spatiale en $\mathcal{O}(4\,\eta^2/M^2)$. De plus, les $2\,M$ fonctions d'agrégation produisent chacune une matrice de similarité, et si cette étape est exécutée sur un seul nœud de calcul, sa complexité spatiale est en $\mathcal{O}(\eta^2)$.

En supposant que l'on dispose de M^2 nœuds pour l'exécution de MVSim, chaque nœud a une complexité spatiale réduite à $4\,(\eta/M)^2 = \mathcal{O}(\eta^2/M^2)$. De plus, comme nous l'avons supposé lors de l'étude de la complexité temporelle de l'approche, on peut également paralléliser les instances de Agg, ce qui nécessite de réutiliser $2\,M$ des M^2 nœuds de calcul. Chacun de ces nœuds doit alors mémoriser $M+1$ matrices de similarités de taille $\eta/M \times \eta/M$, donc leur complexité spatiale pour cette étape est en $\mathcal{O}(\eta^2/M)$. Il est ainsi possible de réduire la mémoire nécessaire à l'étape d'agrégation en augmentant M.

Conclusion relative au découpage selon les deux dimensions

En conclusion, sans découpage et en utilisant χ-Sim pour traiter directement la matrice $\mathbf{R}$, la complexité temporelle est en $\mathcal{O}(\eta^3)$, et la complexité spatiale en $\mathcal{O}(\eta^2)$. Avec cette approche, et en supposant que l'on dispose de M^2 nœuds de calcul, on peut réduire la complexité temporelle à $\mathcal{O}(\eta^3/M^3)$ et la complexité spatiale à $\mathcal{O}(\eta^2/M)$, et fournir une approximation des valeurs de similarités. Ainsi, à l'instar du découpage selon les colonnes uniquement, il est possible de réduire le temps de calcul et la mémoire nécessaire en augmentant M. En revanche, le fait d'augmenter M dégrade certainement la qualité des similarités calculées, et il y a par conséquent un compromis entre rapidité d'exécution et précision des résultats. Comme pour le découpage selon une dimension décrit dans la Sec. III-6.2.a, nous analyserons dans le Chap. IV l'intérêt pratique de cette approche.

> ***Remarque 8.*** *Les nœuds de calcul utilisés dans les approches décrites dans cette section ne doivent pas être sur une unique machine, ce qui annulerait le gain en espace mémoire. De plus, cela créerait une compétition entre les différentes processus pour accéder au bus mémoire.*

III-7 Conclusion du chapitre

Dans ce chapitre, nous avons proposé l'algorithme MVSim qui permet de calculer, à partir d'un jeu de données multivue comportant M matrices de relations, N matrices de similarité pour chacun des types d'objets concerné par les données. Cet algorithme s'appuie sur l'algorithme de calcul de co-similarité χ-Sim présenté dans le Chap. II qui traite une seule matrice de relation. La notion de fonction d'agrégation est également décrite et plusieurs formes sont proposées, permettant notamment de garantir la convergence de l'algorithme MVSim. Par ailleurs, un autre point intéressant de l'algorithme MVSim est qu'il peut être vu comme une généralisation de l'algorithme χ-Sim.

De plus, après avoir étudié la complexité de l'algorithme, nous avons proposé de l'utiliser pour calculer des similarités à partir d'une matrice de relation unique mais de trop grande taille pour être directement traitée par l'algorithme χ-Sim. Plusieurs approches ont été décrites

et peuvent être utilisées en fonction des caractéristiques de la matrice de relation, et de l'application visée.

Chapitre IV

Applications de χ-Sim et de MVSim

Sommaire du chapitre

Dans les chapitres II et III, nous avons décrit les deux principales contributions de cette thèse, à savoir des améliorations de l'algorithme de calcul de co-similarité χ-Sim initialement proposé par Bisson et Hussain (2008), et la définition de l'algorithme multivue de calcul de co-similarité MVSim. *Nous fournissons dans ce chapitre des comparaisons expérimentales avec les méthodes de l'état de l'art présentées dans le Chap. I sur différents jeux de données réels, aussi bien monovues que multivues.*

Dans un premier temps, nous montrerons que les améliorations de l'algorithme χ-Sim permettent effectivement d'accroître sa performance sur un ensemble de jeux de données, et nous étudierons également sa robustesse vis-à-vis de ses différents paramètres.

Dans un second temps, nous comparerons les performances de l'algorithme multivue MVSim *sur des jeux de données multivues, à la fois par rapport à des méthodes monovues (afin de démontrer l'apport d'une approche multivue), et également par rapport à des méthodes multivues. De plus, nous fournirons des exemples d'application de l'algorithme* MVSim *au traitement de matrices de relation de grande taille, grâce à la parallélisation de la méthode, comme présentée dans la Sec. III-6. Deux résultats peuvent être obtenus suivant l'objectif recherché : l'accélération du calcul ou l'amélioration des performances en prenant en compte davantage de données.*

IV-1 Présentation des jeux de données

Dans cette section, nous allons présenter l'ensemble des jeux de données qui seront utilisés dans ce chapitre, à la fois pour évaluer les améliorations apportées à l'algorithme de calcul de co-similarité monovue χ-Sim, mais également pour évaluer les performances de l'algorithme multivue MVSim. En effet, concernant les jeux de données multivues, il est primordial de tester des algorithmes monovues séparément sur les différentes vues, afin de valider l'apport de l'utilisation de toutes les vues. La Tab. 2 résume l'ensemble des jeux de données (6 jeux monovues et 8 jeux multivues) qui seront utilisés dans ce chapitre, en précisant pour chacun d'eux le nombre d'objets et le nombre de relations, ainsi que le nombre de classes de X_1 qui sera toujours le type d'objets que l'on cherche à classifier. De plus, pour certains jeux de données, plusieurs échantillons sont disponibles afin de pouvoir calculer la moyenne et l'écart type des mesures de performance, pour valider la stabilité des méthodes.

> ***Remarque 9.*** *L'intégralité des jeux de données utilisés dans cette thèse, que nous les ayons générés ou non, sont mis à disposition sur internet, afin de pouvoir être ré-utilisés, afin que d'autres chercheurs puissent comparer leurs méthodes aux nôtres. La page où sont disponibles ces jeux de données est* `http://membres-lig.imag.fr/grimal/data.html`.

Nous allons d'abord présenter les jeux de données monovues, puis les jeux de données multivues, qui peuvent être utilisés comme plusieurs jeux de données monovues pour tester l'algorithme χ-Sim, et finalement, nous présenterons un ensemble de jeux de données que nous avons conçu pour pouvoir tester les méthodes de traitement de matrices de grande taille présentées dans la Sec. III-6.2.

IV-1.1 Jeux de données monovues

Les six premiers jeux de données utilisés sont monovues, et ont été créés à partir de la collection de documents textuels de *Newsgroup (NG20)*[8] comportant environ 20.000 articles répartis en 20 groupes, correspondant à 20 sujets de discussions distincts. Cette collection a été utilisée à de multiples reprises pour la validation d'algorithmes d'apprentissage automatique, aussi bien pour des tâches supervisées que non supervisées. Nous avons extrait 6 différents jeux de données (M2, M5, M10, NG1, NG2 et NG3) à partir de cette collection, qui sont classiquement utilisés dans la littérature (Dhillon *et al.* 2003, Long *et al.* 2006a), comportant des nombres de documents et de classes différents, dont les caractéristiques sont décrites dans la Tab.3. Pour chacun de ses jeux, nous avons utilisé 10 échantillons.

Pour les jeux de données de documents que nous avons générés, à savoir M2, M5, M10, NG1, NG2 et NG3, nous avons ignoré les entêtes (qui comporte entre autres la date et le sujet du document) puis éliminé les « mots vides » à partir d'une liste fixée de 571 mots[9], classiquement utilisée pour de telles tâches. Ces mots sont trop courants (*again* et *sometimes* par exemple) pour apporter une quelconque information sur la classe d'un document. Après avoir éliminé ces mots peu informatifs, nous avons effectué une étape de sélection de mots, en

8. Accessible à la page `http://people.csail.mit.edu/jrennie/20Newsgroups/`
9. Accessible à la page `ftp://ftp.cs.cornell.edu/pub/smart/english.stop`

Nom	N (objets)	M (vues)	Echantillons	Classes	Taille
M2	2	1	10	2	X_1 : 500 documents, X_2 : 2000 mots. $V_1 : X_1 \sim X_2$ (co-occurrences)
M5	2	1	10	5	X_1 : 500 documents, X_2 : 2000 mots. $V_1 : X_1 \sim X_2$ (co-occurrences)
M10	2	1	10	10	X_1 : 500 documents, X_2 : 2000 mots. $V_1 : X_1 \sim X_2$ (co-occurrences)
NG1	2	1	10	2	X_1 : 400 documents, X_2 : 2000 mots. $V_1 : X_1 \sim X_2$ (co-occurrences)
NG2	2	1	10	5	X_1 : 1000 documents, X_2 : 2000 mots. $V_1 : X_1 \sim X_2$ (co-occurrences)
NG3	2	1	10	8	X_1 : 1600 documents, X_2 : 2000 mots. $V_1 : X_1 \sim X_2$ (co-occurrences)
IMDb	3	2	1	17	X_1 : 617 films, X_2 : 1878 mots-clés, X_3 : 1398 acteurs. $V_1 : X_1 \sim X_2$ (descripteurs), $V_2 : X_1 \sim X_3$ (casting)
Cora	2	2	1	7	X_1 : 2708 articles, X_2 : 1433 mots. $V_1 : X_1 \sim X_2$ (absence / présence), $V_2 : X_1 \sim X_1$ (citations)
CiteSeer	2	2	1	6	X_1 : 3312 articles, X_2 : 3703 mots. $V_1 : X_1 \sim X_2$ (absence / présence), $V_2 : X_1 \sim X_1$ (citations)
Cornell	2	2	1	5	X_1 : 195 pages web, X_2 : 1703 mots. $V_1 : X_1 \sim X_2$ (co-occurrences), $V_2 : X_1 \sim X_1$ (citations)
Texas	2	2	1	5	X_1 : 187 pages web, X_2 : 1703 mots. $V_1 : X_1 \sim X_2$ (co-occurrences), $V_2 : X_1 \sim X_1$ (citations)
Washington	2	2	1	5	X_1 : 230 pages web, X_2 : 1703 mots. $V_1 : X_1 \sim X_2$ (co-occurrences), $V_2 : X_1 \sim X_1$ (citations)
Wisconsin	2	2	1	5	X_1 : 267 pages web, X_2 : 1703 mots. $V_1 : X_1 \sim X_2$ (co-occurrences), $V_2 : X_1 \sim X_1$ (citations)
ReutersEN	6	5	6	6	X_1 : 1200 documents, X_2 : 2500 mots (anglais), X_3 : 2500 mots (français), X_4 : 2500 mots (allemand), X_5 : 2500 mots (italien), X_6 : 2500 mots (espagnol). $V_1 : X_1 \sim X_2$ (co-occurrences), $V_2 : X_1 \sim X_3$ (traduction), $V_3 : X_1 \sim X_4$ (traduction), $V_4 : X_1 \sim X_5$ (traduction), $V_5 : X_1 \sim X_6$ (traduction)

TABLE 2 – Description de tous les jeux de données – monovues et multivues – utilisés dans le Chap. IV.

	Noms des classes
M2	talk.politics.mideast, talk.politics.misc
M5	comp.graphics, rec.motorcycles, rec.sport.baseball, sci.space, talk.politics.mideast
M10	alt.atheism, comp.sys.mac.hardware, misc.forsale, rec.autos, rec.sport.hockey, sci.crypt, sci.electronics, sci.med, sci.space, talk.politics.gun
NG1	rec.sports.baseball, rec.sports.hockey
NG2	comp.os.ms-windows.misc, comp.windows.x, rec.motorcycles, sci.crypt, sci.space
NG3	comp.os.ms-windows.misc, comp.windows.x, misc.forsale, rec.motorcycles, sci.crypt, sci.space, talk.politics.mideast, talk.religion.misc

Table 3 – Noms des classes des jeux de données issus de NG20.

décidant de ne garder que 2000 mots pour chacun des échantillons, afin de suivre la méthodologie de Dhillon (2001). Comme nous sommes dans un cadre non supervisé, nous ne pouvons pas utiliser l'information sur la classe des documents pour sélectionner les mots les plus discriminants, et avons par conséquent utilisé l'algorithme des k-medoïdes (voir description dans la Sec. I-2.2.a). En effet, cet algorithme réalise un partitionnement de l'espace des mots, afin de trouver k représentants, ce qui est une méthode raisonnable dans notre cas, car cela permet de sélectionner des mots relativement indépendants les uns des autres.

Finalement, le nombre de types d'objets est de 2, avec X_1 désignant les documents et X_2 les mots. Précisons que le modèle de pondération (voir Sec. I-1.1.b) choisi pour ces jeux de données est le nombre de co-occurrences des mots dans les documents, qui sera donc représenté par l'unique matrice de relation $\mathbf{R}_1$.

IV-1.2 Jeux de données multivues

Nous avons utilisé huit jeux de données multivues, dont le nombre de vues N varie entre 2 et 5, et le nombre d'objets M varie entre 2 et 6.

Le premier jeux de données multivue a été extrait à partir du site internet *IMDb* (de l'anglais *Internet Movie Database*), un site proposant une base de données très complète sur les films [10]. Nous avons cherché à classifier automatiquement les films en fonction de leur genre (action, comédies, etc.), et avons pour cela créer deux vues pour les décrire. La première vue V_1 décrit la relation entre les films et les acteurs qui y jouent, ce qui fournit une matrice binaire $\mathbf{R}_1$. Puis, nous avons extrait une seconde vue V_2 décrivant la relation entre les films et des mots-clés utilisés par les utilisateurs du site *IMDb* pour les décrire, ce qui fournit une matrice de co-occurrences $\mathbf{R}_2$. Nous avons alors trois types d'objets : X_1 désigne les films, X_2 désigne les acteurs et X_3 désigne les mots-clés. Les 17 genres des films, qui représentent donc les classes de X_1 que l'on cherche à apprendre sont donnés dans la Tab. 4, et sont relativement équilibrés (moyenne de 36 films par genre, avec un écart-type de 9).

10. Accessible à la page http://www.imdb.com/interfaces/

	Nom des classes
IMDb	short, drama, comedy, documentary, romance, action, animation, thriller, crime, family, adventure, horror, fantasy, mystery, western, sci-fi, biography

Table 4 – Noms des classes du jeu de données multivue *IMDb*.

Les 6 jeux de données multivues suivants[11] (Cora, CiteSeer, Cornell, Texas, Washington et Wisconsin) ont des structures similaires, à savoir qu'ils concernent deux types d'objets : X_1 qui désigne des documents (articles scientifiques pour Cora et CiteSeer, et pages internet pour les quatre autres) et X_2 qui désigne des mots. De plus, ils contiennent tous deux vues décrits par une matrice documents - documents $\mathbf{R}_1$ et une matrice documents - mots $\mathbf{R}_2$. La matrice documents - documents représente une relation de citation dans le cas des articles scientifiques et une relation de lien hypertexte dans le cas des pages internet. Il est ainsi possible de prendre en compte le sens des liens (ou des citations) en utilisant cette matrice tel quel ou en considérant sa transposée $\mathbf{R}_1^\top$; ou bien de l'ignorer en construisant la matrice $\mathbf{R}_1 + \mathbf{R}_1^\top$. Concernant les six matrices documents - mots, le modèle de pondération adopté par les auteurs de ces jeux de données est le modèle binaire (absence ou présence des mots dans les documents). Les classes des documents sont indiquées dans la Tab. 5.

	Nom des classes
Cora	neural networks, rule learning, reinforcement learning, probabilistic methods, theory, genetic algorithms, case based
CiteSeer	agents, information retrieval, databases, artificial intelligence, human-computer interaction, machine learning
Cornell, Texas, Washington, Wisconsin	student, project, course, staff, faculty

Table 5 – Noms des classes des jeux de données multivues Cora, CiteSeer, Texas, Washington et Wisconsin.

Finalement, nous avons construit un jeu de données à partir de la collection multilingue de documents provenant de l'agence de presse *Reuters*[12], créée par Amini *et al.* (2010). Suivant la méthodologie de Kumar et Daume III (2011), nous avons extrait 1200 documents, représentés par X_1 rédigés en anglais, et avons donc extrait une première vue V_1 contenant les co-occurrences des mots anglais, représentés par X_2. Puis nous avons utilisé les traductions de ces documents dans quatre autres langues pour créer quatre vues supplémentaires sur les données. Nous obtenons ainsi en plus de la matrice $\mathbf{R}_1$ des mots anglais, les matrices $\mathbf{R}_2$, $\mathbf{R}_3$, $\mathbf{R}_4$ et $\mathbf{R}_5$, représentant respectivement les traductions des documents en français, allemand, italien et espagnol. De plus, nous avons répété le procédé six fois, afin d'obtenir 6 échantillons. Comme pour la création des jeux de données monovues issus de la collection NG20, nous avons, pour chacune de ses vues, sélectionné 2000 mots en utilisant l'algorithme des k-medoïdes. Les classes des documents sont données dans la Tab. 6, et correspondent à des catégories d'articles.

11. Accessible à la page http://www.cs.umd.edu/projects/linqs/projects/lbc/
12. Accessible à la page http://multilingreuters.iit.nrc.ca/ReutersMultiLingualMultiView.htm

	Nom des classes
ReutersEN	E21, CCAT, M11, GCAT, C15, ECAT

Table 6 – Noms des classes du jeu de données multivue ReutersEN.

Rappelons pour finir, que ces jeux de données, bien que multivues, seront également utilisés pour tester l'algorithme χ-Sim où chaque vue sera traitée comme un jeu de données indépendant. En plus d'agrandir la collection de jeux de données sur lesquels nous allons comparer l'algorithme χ-Sim aux autres méthodes, cela fournira également une performance à laquelle on pourra comparer l'ensemble des méthodes multivues.

IV-1.3 Jeux de données monovues de grande taille pour MVSim

Finalement, afin de valider les approches de découpage de matrices de relation de grande taille présentées dans la Sec. III-6.2, nous avons construit une famille de jeux de données issus de la collection de documents NG20 également [13]. Contrairement aux jeux de données monovues décrits dans la Sec. IV-1.1, nous avons fixé le nombre de classes à 10 (données dans la Tab. 7), et nous avons fait varier à la fois le nombre de documents sélectionnés par classe, et également le nombre de mots sélectionnés en utilisant toujours l'algorithme des k-medoïdes. Ainsi, nous disposons d'une famille de jeux de données homogènes, mais ayant des tailles différentes.

	Nom des classes
NG20-split	comp.graphics, misc.forsale, rec.autos, rec.sport.baseball, rec.sport.hockey, sci.crypt, sci.med, sci.space, soc.religion.christian, talk.politics.mideast

Table 7 – Noms des classes de la famille de jeux de données NG20-split, utilisée pour l'application de MVSim à une matrice de grande taille.

Le nombre de documents par classe est parmi $[\![40, 80, 160, 320, 640]\!]$, ce qui donne un total de documents entre 400 et 6400. Le nombre de mots est quant à lui parmi $[\![250, 500, 1000, 2000, 4000]\!]$. De plus, pour chacun de ces jeux de données, nous avons construit 10 échantillons, afin de pouvoir calculer la moyenne et l'écart-type des mesures d'évaluation.

IV-2 Application de χ-Sim à la classification

Dans cette section, nous allons utiliser l'algorithme de calcul de co-similarité χ-Sim pour calculer la matrice de similarité $\mathbf{S}_1$, qui contient les similarités entre toutes les paires des instances de X_1 pour tous les jeux de données décrits dans la Tab. 2. Généralement X_1 représentera des documents textuels (articles scientifiques, pages internet, etc.), mais également des films dans le cas du jeu de données *IMDb*. Nous comparerons l'algorithme à plusieurs autres méthodes de l'état de l'art présentées dans le Chap. I. Nous étudierons de plus la sensibilité des performances de l'algorithme χ-Sim par rapport à ces différents paramètres.

13. Accessible à la page http://people.csail.mit.edu/jrennie/20Newsgroups/

IV-2.1 Méthodologie

Comme nous l'avons précisé dans l'introduction de la Sec. IV-1, le type d'objets que nous cherchons à classifier est toujours X_1, pour lequel nous disposons, pour chacune de ses instances, d'une étiquette correspondant à sa classe réelle. Ainsi, nous allons classifier les instances de X_1, à partir de différentes méthodes – dont χ-Sim – et comparer les classes prédites avec les classes réelles de ces instances, à l'aide des différentes mesures présentées dans la Sec. I-5. Nous privilégierons l'utilisation de la précision micro-moyennée (Pr) car c'est celle qui est la plus intuitive, et les autres mesures seront utilisées ponctuellement, lorsque nous ne disposons pas de l'implémentation d'une méthode et que nous citons simplement le résultat qu'elle obtient. Il est également important de préciser que dans la plupart des cas, les comparaisons des performances de différentes méthodes sont indépendantes de la mesure utilisée.

Les méthodes auxquelles nous allons comparer l'algorithme χ-Sim sont les suivantes :

- la mesure de similarité du Cosinus,
- la mesure LSA (de l'anglais *Latent Semantic Analysis*) de Deerwester *et al.* (1990),
- la mesure SNOS (de l'anglais *Similarity in Non-Orthogonal Spaces*) de Liu *et al.* (2004),
- la mesure CTK (de l'anglais *Commute-Time Kernel*) de Yen *et al.* (2009),
- l'algorithme ITCC (de l'anglais *Information Theory Co-Clustering*) de Dhillon (2001),
- la mesure issue de l'algorithme χ-Sim original de Bisson et Hussain (2008).

Excepté SNOS, ces méthodes ont toutes été présentées dans le Chap. I, et dans le Chap. II pour l'algorithme χ-Sim original. La mesure SNOS a été développée dans le cadre de la catégorisation de documents (apprentissage supervisé) et son concept de base est proche de celui de χ-Sim. En effet, la mesure SNOS définit la similarité des documents par rapport à la similarité entre les mots qui les composent, et inversement. La différence majeure entre cette mesure et χ-Sim tient à sa normalisation, qui a un impact majeur sur la performance.

Pour toutes les mesures de similarité citées (seul ITCC est un algorithme qui classifie directement les données), nous disposons alors d'une matrice de similarité $\mathbf{S}_1$, et nous devons donc utiliser un algorithme de classification, prenant en argument $\mathbf{S}_1$, afin de prédire la classification des instances de X_1. Le choix est ensuite relativement peu important car, à partir du moment où l'algorithme utilisé est le même pour toutes les mesures de similarité, les résultats seront *a priori* comparables. Nous avons alors choisi d'utiliser l'algorithme de Classification Ascendante Hiérarchique (CAH) avec le critère de Ward, présenté dans la Sec. I-2.2.b, et qui produit un dendrogramme. Puis, nous coupons ce dendrogramme au niveau nous permettant d'obtenir le nombre de classes désirées. En pratique, l'algorithme de CAH prend en argument une matrice de distance et nous devons donc transformer notre matrice de similarité $\mathbf{S}_1$ en matrice de distance. Nous faisons alors le choix simple de définir la distance $d(x_i^{(1)},x_j^{(1)})$ entre deux instances de X_1 comme étant $1 - s_{ij}^{(1)}$.

L'algorithme χ-Sim a été implémenté en Python, en utilisant la bibliothèque de calcul scientifique SciPy (Jones *et al.* 2001), tout comme la similarité du Cosinus et la mesure SNOS. Pour la mesure CTK et l'algorithme ITCC, nous avons utilisé les implémentations fournies par les auteurs (Yen *et al.* 2009, Dhillon 2001). Nous avons enfin utilisé l'implémentation issue

du module python *mlpy* (Albanese *et al.* 2012), dédié à l'apprentissage automatique, pour la Classification Ascendante Hiérarchique [14].

Pour ces différentes mesures et algorithmes, leurs paramètres doivent être réglés au mieux, afin d'obtenir une comparaison équitable. Dans un contexte d'apprentissage non supervisé, ces paramètres ne peuvent pas être directement appris à partir d'un jeu de données de validation. Pour cela, nous avons suivi les recommandations des auteurs de ces méthodes, et nous rapporterons dans la section suivante uniquement le meilleur résultat obtenu :

- **LSA :** l'unique paramètre de l'algorithme LSA est le nombre de valeurs singulières conservées s, qui permettent de calculer une approximation de rang s de la matrice de relation. Nous avons fait varier s entre 10 et 200, avec un pas de 10.
- **SNOS :** les deux paramètres de cet algorithme sont le nombre d'itérations et le paramètre de normalisation. Nous avons fixé le nombre d'itérations à 8 (les auteurs observent que leur algorithme converge après 8 itérations en moyenne) et le paramètre de normalisation à la valeur recommandée par les auteurs (Liu *et al.* 2004).
- **CTK :** le seul paramètre à fixer est le facteur multiplicateur α, et nous avons utilisé la valeur fournissant les meilleurs résultats dans les expérimentations des auteurs sur des jeux de données issus de la collection NG20, à savoir $\alpha = 7$.
- **ITCC :** le paramètre qui doit être fixé pour l'algorithme ITCC est le nombre de classes des objets décrits par les colonnes (les mots classiquement), et nous avons donc fait varier ce paramètre comme le recommandent Dhillon (2001), en testant avec 32, 64 et 128 classes. De plus, l'initialisation de l'algorithme est aléatoire, la procédure est donc répétée trois fois.
- **χ-Sim original :** l'unique paramètre de la version originale de l'algorithme χ-Sim est le nombre d'itérations T, et nous l'avons fixé à 4.

Concernant la nouvelle version de l'algorithme χ-Sim, les paramètres à faire varier sont le nombre d'itérations T, le paramètre de seuillage p, et le paramètre de la pseudo-norme k. Comme pour la version originale de l'algorithme, nous fixons le nombre d'itérations à 4. Puis, nous avons fait varier le paramètre d'élagage p entre 0 et 0,8 avec un pas de 0,2, et le paramètre de pseudo-norme k entre 0,6 et 1,2 avec un pas de 0,2 également.

IV-2.2 Résultats

L'ensemble des résultats expérimentaux sont présentés dans la Tab. 8. Nous avons testé les méthodes mentionnées dans la section précédente, sur tous les jeux de données décrits dans la Sec. IV-1, et nous rapportons le meilleur résultat en terme de précision micro-moyennée.

Sur les 25 jeux de données testés, nous remarquons que χ-Sim obtient le meilleur résultat pour 15 d'entre eux, soit 60%. Pour les 10 jeux restants, soit LSA soit CTK obtient un meilleur résultat que χ-Sim, qui obtient alors systématiquement le second meilleur résultat. D'un point de vue global, les performances de l'algorithme χ-Sim sont donc tout à fait satisfaisantes.

14. Ce module est accessible à la page `http://mlpy.sourceforge.net/`

Données	Cosinus	LSA	SNOS	CTK	ITCC	χ-Sim original	χ-Sim
M2	0,61 ± 0,04	0,79 ± 0,09	0,51 ± 0,01	0,67 ± 0,10	0,70 ± 0,05	0,58 ± 0,07	**0,81** ± 0,10
M5	0,54 ± 0,08	0,66 ± 0,05	0,26 ± 0,04	0,76 ± 0,04	0,54 ± 0,05	0,62 ± 0,12	**0,79** ± 0,05
M10	0,39 ± 0,03	0,44 ± 0,04	0,20 ± 0,02	0,54 ± 0,05	0,29 ± 0,05	0,43 ± 0,04	**0,55** ± 0,04
NG1	0,52 ± 0,01	0,56 ± 0,05	0,51 ± 0,00	0,69 ± 0,14	0,61 ± 0,06	0,54 ± 0,03	**0,81** ± 0,02
NG2	0,60 ± 0,05	0,61 ± 0,06	0,24 ± 0,01	0,64 ± 0,06	0,44 ± 0,08	0,60 ± 0,12	**0,73** ± 0,06
NG3	0,49 ± 0,02	0,52 ± 0,03	0,22 ± 0,02	0,54 ± 0,02	0,49 ± 0,07	0,47 ± 0,05	**0,64** ± 0,04
IMDb (V_1)	0,225	0,277	0,088	0,225	0,280	0,288	**0,290**
IMDb (V_2)	0,212	0,216	0,083	0,214	0,160	0,222	**0,240**
Cora (V_1)	0,304	**0,497**	0,358	0,234	0,250	0,425	0,473
Cora (V_2)	0,305	0,303	0,266	0,186	0,401	0,254	**0,502**
CiteSeer (V_1)	0,219	**0,317**	0,216	0,212	0,227	0,284	0,305
CiteSeer (V_2)	0,211	0,463	0,328	0,246	0,532	0,500	**0,608**
Cornell (V_1)	0,333	0,364	0,421	**0,461**	0,385	0,318	0,436
Cornell (V_2)	0,436	0,579	0,436	0,410	0,477	0,456	**0,631**
Texas (V_1)	0,422	0,476	0,417	0,524	0,492	0,385	**0,535**
Texas (V_2)	0,439	0,663	0,551	0,304	0,396	0,561	**0,722**
Washington (V_1)	0,417	0,517	0,417	0,465	0,391	0,357	**0,548**
Washington (V_2)	0,478	**0,652**	0,478	0,530	0,551	0,583	0,617
Wisconsin (V_1)	0,347	0,411	0,381	**0,434**	0,358	0,366	0,419
Wisconsin (V_2)	0,332	0,604	0,468	0,347	0,581	0,540	**0,675**
ReutersEN (V_1)	0,357 ± 0,025	**0,582** ± 0,048	0,294 ± 0,012	0,350 ± 0,044	0,432 ± 0,018	0,253 ± 0,010	0,477 ± 0,012
ReutersEN (V_2)	0,351 ± 0,015	**0,601** ± 0,031	0,297 ± 0,020	0,339 ± 0,065	0,430 ± 0,015	0,254 ± 0,015	0,474 ± 0,016
ReutersEN (V_3)	0,349 ± 0,012	**0,585** ± 0,027	0,294 ± 0,012	0,306 ± 0,065	0,430 ± 0,031	0,248 ± 0,011	0,467 ± 0,014
ReutersEN (V_4)	0,311 ± 0,062	**0,600** ± 0,049	0,295 ± 0,011	0,378 ± 0,019	0,431 ± 0,026	0,253 ± 0,012	0,464 ± 0,025
ReutersEN (V_5)	0,318 ± 0,064	**0,591** ± 0,035	0,300 ± 0,013	0,316 ± 0,061	0,441 ± 0,055	0,262 ± 0,014	0,492 ± 0,028

TABLE 8 – Résultats pour les méthodes monovues, exprimés en précision micro-moyennée (Pr).

En regardant plus en détail, nous notons que les améliorations apportées à l'algorithme χ-Sim lui permettent de significativement améliorer sa performance, car pour les 25 jeux de données, le résultat de l'algorithme χ-Sim original est inférieur à celui de la nouvelle version proposée. Ce résultat démontre l'intérêt du nouveau schéma de normalisation proposé, associé à la pseudo-norme k et à l'étape de seuillage.

De plus, nous remarquons que la mesure de similarité SNOS obtient de médiocres résultats bien que comme nous l'ayons mentionné plus tôt, elle soit très proche sur le principe, de l'algorithme χ-Sim. La différence principale étant la normalisation choisie, cela confirme que le schéma de normalisation joue un rôle primordial dans la définition de la mesure de similarité, et que s'il n'est pas judicieusement choisi, il peut considérablement détériorer les performances d'une telle mesure.

Pour compléter cette analyse comparative, nous avons calculé le rang moyen de chaque méthode sur ces 25 jeux de données. Pour chaque jeu (chaque ligne de la Tab. 8), nous avons ordonné les scores des différentes méthodes du plus grand au plus petit, et le rang correspond alors à la position de la méthode : celle qui obtient le meilleur résultat aura un rang de 1, et celle qui aura le moins bon un rang de 7. La Tab. 9 présente les rangs moyens (associé à leur écart-type) des méthodes utilisées sur les 25 jeux de données.

Cosinus	LSA	SNOS	CTK	ITCC	χ-Sim original	χ-Sim
$5{,}0 \pm 1{,}0$	$2{,}5 \pm 1{,}3$	$5{,}8 \pm 1{,}2$	$4{,}3 \pm 2{,}0$	$4{,}0 \pm 1{,}4$	$4{,}9 \pm 1{,}7$	$\mathbf{1{,}4 \pm 0{,}5}$

Table 9 – Rangs moyens des méthodes comparées, calculés à partir de la Tab. 8.

Grâce à cette analyse des rangs moyens, nous obtenons bien la confirmation que la nouvelle version de l'algorithme χ-Sim obtient en moyenne la meilleure performance, avec LSA en seconde position. Il est intéressant de remarquer que l'algorithme CTK, bien qu'obtenant le meilleur résultat sur 2 des 25 jeux de données, obtient un rang moyen assez élevé car ses performances sur les autres jeux de données sont plus faibles. Cette variabilité de la performance de l'algorithme CTK se retrouve d'ailleurs dans le grand écart-type de ses rangs moyens dans la Tab. 9.

De plus, cela nous sera utile dans la partie suivante, il est intéressant d'observer quelle méthode obtient le meilleur résultat toutes vues confondues pour chaque jeu de données. La nouvelle version de l'algorithme χ-Sim obtient ainsi le meilleur résultat pour 6 des 8 jeux de données multivues, et le second meilleur résultat dans les 2 restants (Washington et ReutersEN), pour lesquels LSA obtient le meilleur score.

En conclusion de cette section, et pour nuancer cette analyse de la performance de l'algorithme χ-Sim, notons que nous avons pu le tester en faisant varier ses paramètres dans des intervalles que nous savons intéressants, alors que pour certaines méthodes – SNOS et CTK en particulier – nous disposons de peu d'information sur la détermination des paramètres, et

nous les avons donc peu ou pas fait varier. Il est alors possible qu'en re-testant ces méthodes avec plus de valeurs pour leurs paramètres, leurs performances s'améliorent.

IV-2.3 Sensibilité aux paramètres

Dans cette section, nous allons étudier la sensibilité de l'algorithme χ-Sim à ces trois paramètres :

- le nombre d'itérations T, que nous allons faire varier entre 1 et 10 avec un pas de 1 ;
- le paramètre de pseudo-norme k, que nous allons faire varier entre 0,6 et 1,2 avec un pas de 0,2 ;
- le paramètre de seuillage p, que nous allons faire varier entre 0 et 0,8 avec un pas de 0,2.

Pour valider la robustesse de l'algorithme vis-à-vis de ces paramètres, nous allons réaliser 3 graphes permettant chacun d'analyser les variations de performances en terme de précision micro-moyennée de l'algorithme χ-Sim. Afin de ne pas surcharger les graphes, nous ne tracerons les courbes que pour quelques jeux de données représentatifs, sachant que les variations sont semblables pour les autres. Chaque point sera donc la moyenne de la précision micro-moyennée pour un jeu de données, et les barres d'erreur représentent les écarts-types. La moyenne et l'écart-type pour chaque point est calculée à partir des différents échantillons du jeu, et aussi des différentes valeurs des autres paramètres.

La Fig. 22 représente la variation de précision micro-moyennée en fonction du nombre d'itérations T de l'algorithme χ-Sim. On observe une nette augmentation pendant les 3 premières itérations, puis une lente dégradation jusqu'à la dixième itération de l'algorithme. On note que ce comportement est similaire pour tous les jeux de données observés, et cela confirme notre choix de ne pas laisser l'algorithme aller au delà de la quatrième itération dans nos expérimentations.

La Fig. 23 représente la variation de précision micro-moyennée en fonction du paramètre de pseudo-norme k de l'algorithme χ-Sim. On observe que la performance de l'algorithme est très stable vis-à-vis de ce paramètre, ce qui démontre la robustesse de la méthode. On remarque que même si la propriété de semi-définie positivité des matrices de similarités qui sont calculées par l'algorithme est uniquement garantie pour $k = 1$, les performances ne sont pas dégradées par des valeurs de k inférieures à 1. En effet, nous rappelons que c'est l'observation du bon comportement de la norme L_k par Aggarwal *et al.* (2001) dans des espaces à grandes dimensions pour des valeurs de k plus petites que 1 qui a inspiré l'introduction de cette pseudo-norme. La valeur par défaut que nous utiliserons sera $k = 0{,}8$.

La Fig. 24 représente la variation de précision micro-moyennée en fonction du paramètre de seuillage p de l'algorithme χ-Sim. On observe que la performance de l'algorithme est relativement stable vis-à-vis de ce paramètre, ce qui démontre également la robustesse de la méthode, même si l'algorithme est plus sensible à ce paramètre qu'au paramètre de pseudo-norme k. C'est en moyenne pour les valeurs les plus fortes, à savoir 0,6 et 0,8, que les performances de χ-Sim sont les meilleurs, et c'est 0,6 que nous considérons comme la valeur par défaut

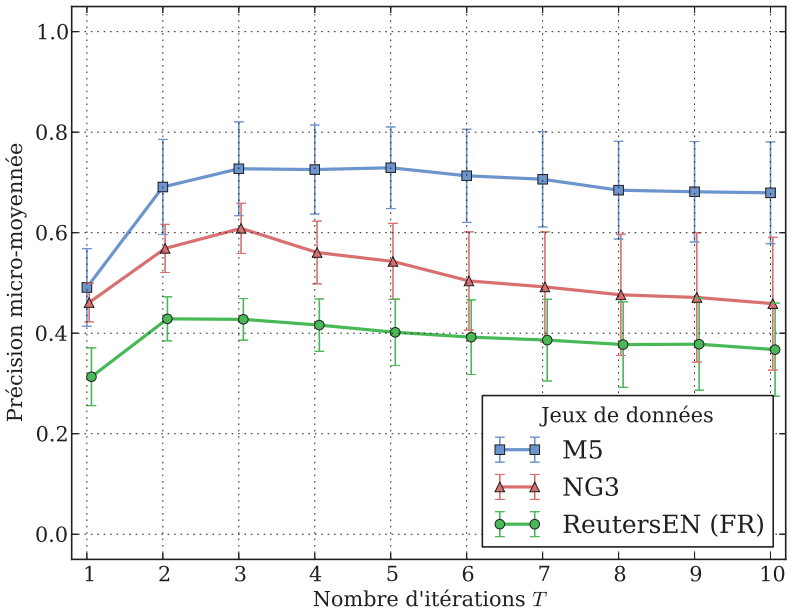


Figure 22 – Variations des performances de χ-Sim en fonction du nombre d'itérations T. Les valeurs sont les moyennes (et écarts-types) des précisions micro-moyennées sur les 10 échantillons et pour $k \in [0{,}6; 0{,}8; 1; 1{,}2]$ et $p \in [0; 0{,}2; 0{,}4; 0{,}6; 0{,}8]$.

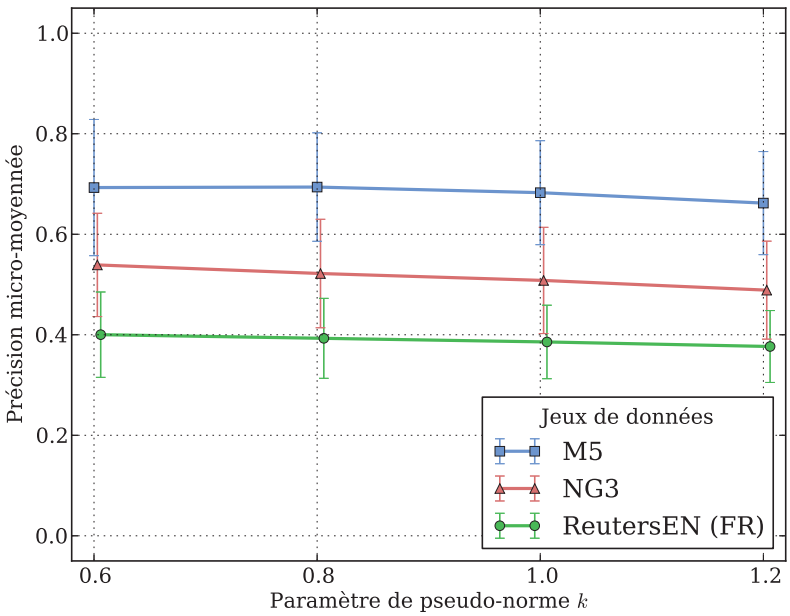


Figure 23 – Variations des performances de χ-Sim en fonction de la pseudo-norme k. Les valeurs sont les moyennes (et écarts-types) des précisions micro-moyennées sur les 10 échantillons et pour $t \in [1; 2; 3; 4; 5; 6; 7; 8; 9; 10]$ et $p \in [0; 0{,}2; 0{,}4; 0{,}6; 0{,}8]$.

pour le paramètre de seuillage p. Le fait que ce soit pour ces valeurs que les résultats soient les meilleurs semblent indiqué que l'algorithme bénéficie grandement de la mise à zéro d'une grande proportion des valeurs de similarités qu'il calcule.

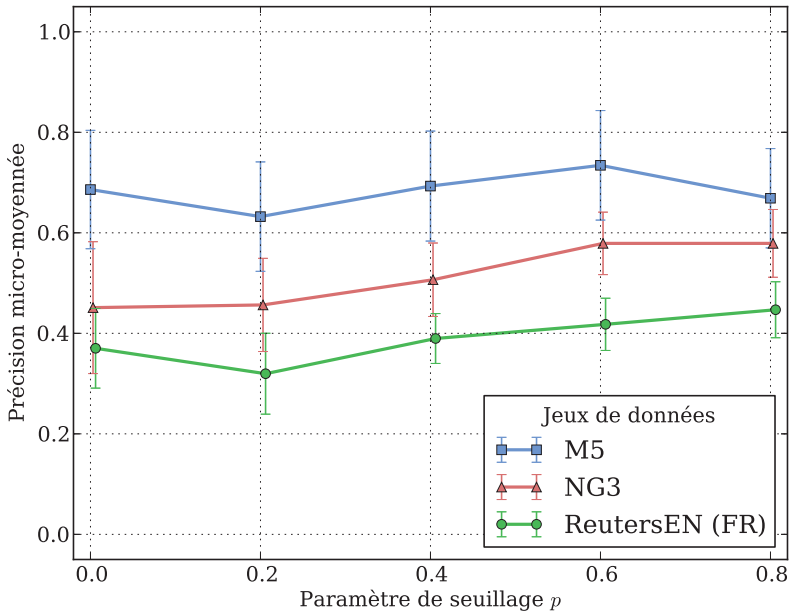


Figure 24 – Variations des performances de χ-Sim en fonction du paramètre de seuillage p. Les valeurs sont les moyennes (et écarts-types) des précisions micro-moyennées sur les 10 échantillons et pour $t \in [1; 2; 3; 4; 5; 6; 7; 8; 9; 10]$ et $k \in [0{,}6; 0{,}8; 1; 1{,}2]$.

En conclusion de cette étude de la sensibilité de l'algorithme χ-Sim à ces paramètres T, k et p, on peut dire que ces performances sont stables pour différentes valeurs de ces paramètres, et qu'il est donc relativement facile de l'appliquer à des tâches d'apprentissage non supervisé pour lesquelles il est délicat de choisir les paramètres. L'Annexe B présente plusieurs graphes 3D de variations des performances de χ-Sim permettant de compléter cette analyse.

IV-3 Application de MVSim à la classification multivue

Dans cette section, nous allons utiliser l'algorithme de calcul multivue de co-similarité MVSim pour calculer les matrices de similarité des différents types d'objets, pour tous les jeux de données multivues décrits dans la Sect. IV-1.2. Pour chacun de ces jeux de données, il y a donc plusieurs types d'objets, et nous calculerons donc une matrice de similarité pour chacun de ces types, mais le seul type d'objets pour lequel nous disposons de classes réelles sera toujours X_1. Nous allons comparer notre algorithme aux méthodes monovues déjà utilisées dans la section précédente, et également à des méthodes multivues présentées dans le Chap. I. Nous étudierons de plus la sensibilité des performances de l'algorithme MVSim par rapport à ces différents paramètres.

IV-3.1 Méthodologie

Comme nous l'avons dit précédemment, nous souhaitons commencer par valider le fait que l'apprentissage multivue des similarités permet d'améliorer la classification des instances de X_1, en comparant les résultats de l'algorithme MVSim aux résultats des méthodes monovues présentées dans la section précédente. Sachant que nous avons déjà donné le détail des résultats de ces méthodes dans la Tab. 8, nous ne reporterons dans cette section à titre de comparaison uniquement le meilleur score, pour toutes les méthodes et toutes les vues d'un jeu donné.

Au delà de la validation de l'approche multivue par rapport aux méthodes monovues, nous souhaitons également comparer l'algorithme MVSim à des méthodes également capables d'exploiter les multiples vues de ces jeux de données. La première méthode que nous utiliserons est :

- l'algorithme MVSC (de l'anglais *Multi-View Spectral Clustering*) de Kumar et Daume III (2011)

De plus, nous utiliserons trois versions « naïves » de l'algorithme MVSim, pour lesquelles $T_{\text{MV}} = 1$ afin de valider la proposition que le fait de faire les itérations au niveau global permet un partage de l'information entre les vues :

- la mesure Cosinus MV est une simple moyenne des mesures de Cosinus, et nous l'a considérons comme une similarité du Cosinus multivue. Elle correspond à MVSim avec $k = 1$, $p = 0$, $\alpha(t) = 1$ et F la moyenne.
- la mesure χ-Sim+Moyenne représente un calcul local (au niveau des vues) de co-similarités, avec une agrégation seulement à la fin. Elle correspond à MVSim avec $T = 4$, $\alpha(t) = 1$ et F la moyenne.
- la mesure χ-Sim+CSPA représente un calcul local (au niveau des vues) de co-similarités, avec une agrégation basée sur une méthode de consensus de classification. Elle correspond à MVSim avec $T = 4$, $\alpha(t) = 1$ et F la fonction CSPA.

Pour les deux dernières méthodes, les valeurs pour les paramètres k et p varient dans les mêmes intervalles que dans la Sec. IV-2. Notons que le nombre d'itérations T des instances de l'algorithme χ-Sim n'est d'ordinaire pas considéré comme un paramètre de MVSim car il est fixé à 1, et les itérations se font au niveau global. La méthode χ-Sim+Moyenne représente une extension naïve de l'algorithme χ-Sim aux données multivues, et la méthode χ-Sim+CSPA fournit une comparaison avec une méthode de consensus de classification.

Ces trois méthodes nous fournirons donc une comparaison de base, car elles sont des versions naïves de l'algorithme MVSim, qui n'utilisent pas la fonction d'agrégation Agg, et pas la réinitialisation des instances de χ-Sim.

Par ailleurs, les auteurs de l'extension de l'algorithme des k-moyennes aux données multivues (Drost *et al.* 2006) n'ont pas été en mesure de nous fournir une implémentation de leur algorithme, mais ont publié les résultats de leur méthode sur le jeu de données CiteSeer. Ainsi, nous ne pourrons pas tester cet algorithme – que nous désignerons par MVKM dans la suite – sur les 8 jeux de données multivues dont nous disposons, mais nous comparerons ses résultats sur CiteSeer avec MVSim.

A l'instar des mesures de similarités monovues utilisées dans la section précédente, nous utiliserons pour les mesures Cosinus MV, χ-Sim+Moyenne et χ-Sim+CSPA, ainsi que pour l'algorithme MVSim l'algorithme de Classification Ascendante Hiérarchique (CAH) avec le critère de Ward (présenté dans la Sec. I-2.2.b), pour construire un dendrogramme, que nous couperons au niveau nous fournissant le nombre de classes désiré. La matrice de similarité $\mathbf{S}_1$ sera alors pour cela transformée en matrice de distance en utilisant la transformation simple suivante : la distance $d(x_i^{(1)},x_j^{(1)})$ entre deux instances de X_1 est définie comme $1 - s_{ij}^{(1)}$.

Comme l'algorithme χ-Sim, nous avons implémenté l'algorithme MVSim en Python, en utilisant la bibliothèque de calcul scientifique SciPy (Jones *et al.* 2001). Quant à l'algorithme MVSC, nous avons utilisé l'implémentation fournie par ses auteurs (Kumar et Daume III 2011). Comme pour les mesures de similarité monovues, nous avons utilisé l'implémentation de la CAH du module python *mlpy* (Albanese *et al.* 2012), dédié à l'apprentissage automatique [15].

Concernant l'algorithme MVSC, nous avons utilisé les paramètres utilisés par défaut par les auteurs dans l'implémentation qu'ils nous ont fournie, à savoir que le nombre d'itérations a été fixé à 5, et le nombre de vecteurs propres utilisés pour la projection à 1,5. Précisons également que l'algorithme ne fournit alors pas une seule classification par jeu de données, mais une classification par itération et par vue. Ainsi, nous reporterons le meilleur résultat obtenu, toutes itérations et toutes vues confondues.

Rappelons ici que les paramètres de l'algorithme MVSim sont :

- le nombre d'itérations T_{MV} que nous fixons à 10 ;
- le paramètre de pseudo-norme k des instances de χ-Sim que nous avons fait varier entre 0,6 et 1,2 avec un pas de 0,2 ;
- le paramètre de seuillage p des instances de χ-Sim que nous avons fait varier entre 0 et 0,8 avec un pas de 0,2 ;
- la fonction d'agrégation élémentaire F, où nous avons utilisé le Minimum, le Maximum, la Moyenne et la fonction CSPA ;
- le paramètre $\alpha(t)$ des fonction d'agrégation que nous avons fixé soit à λ^{t-1}, soit à $\frac{\lambda^{t-1}}{1+\lambda^{t-1}}$.

De la même façon que dans la section précédente, nous allons privilégier la précision micro-moyennée (Pr) pour comparer les performances de ses différentes méthodes, par rapport à l'information mutuelle normalisée (NMI) et l'entropie (H) que nous utiliserons dans le cas où nous citons les résultats de MVKM dont nous n'avons pas l'implémentation.

IV-3.2 Résultats

L'ensemble des résultats expérimentaux pour les jeux de données multivues décrits dans la Sec. IV-1.2 sont présentés dans la Tab. 10.

Sur les 8 jeux de données multivues testés, l'algorithme MVSim obtient les meilleurs résultats pour 6 d'entre eux, soit 75%. Pour les 2 jeux restants, qui sont Texas et ReutersEN, des méthodes monovues obtiennent la meilleure performance, respectivement χ-Sim et LSA.

15. Ce module est accessible à la page `http://mlpy.sourceforge.net/`

Données	Meilleur monovue	Cosinus MV	χ-Sim +Moyenne	χ-Sim +CSPA	MVSC	MVSim
IMDb	0,290 (χ-Sim, V_1)	0,191	0,298	0,293	0,296	**0,347**
Cora	0,502 (χ-Sim, V_2)	0,394	0,550	0,526	0,528	**0,697**
CiteSeer	0,608 (χ-Sim, V_2)	0,405	0,582	0,573	0,578	**0,635**
Cornell	0,631 (χ-Sim, V_2)	0,364	0,595	0,554	0,519	**0,708**
Texas	**0,722** (χ-Sim, V_2)	0,497	0,594	0,487	0,591	0,679
Washington	0,652 (LSA, V_2)	0,470	0,582	0,526	0,605	**0,709**
Wisconsin	0,675 (χ-Sim, V_2)	0,600	0,604	0,566	0,551	**0,706**
ReutersEN	**0,601** $\pm$ 0,031 (LSA, V_2)	0,357 $\pm$ 0,025	0,473 $\pm$ 0,023	0,492 $\pm$ 0.027	0,510 $\pm$ 0,017	0,509 $\pm$ 0,056

Table 10 – Résultats des méthodes multivues, exprimés en précision micro-moyennée (Pr).

Il est surprenant de voir des méthodes monovues obtenir de meilleurs performances que les méthodes multivues pour ces jeux de données. Cependant, en observant les performances de l'algorithme χ-Sim sur les 2 vues du jeu de données Texas dans la Tab. 8, on s'aperçoit que la précision obtenue à partir de la première vue (liens entre pages internet) est de 0,535 alors qu'elle est de 0,722 à partir de la seconde vue (co-occurrences des mots dans les pages). Ceci semble indiquer que la seconde vue est beaucoup plus informative que la première, et que les approches multivues ne sont pas capables dans ce cas d'agréger correctement les informations provenant de ces 2 vues. Concernant le jeu de données ReutersEN, rappelons qu'il s'agissait d'un des rares jeux où la méthode LSA obtenait de meilleurs résultats que l'algorithme χ-Sim. Sachant que l'algorithme MVSim est basé sur χ-Sim, il n'est alors pas étonnant de voir qu'il échoue également sur ce jeu de données.

Par ailleurs, nous avions construit trois méthodes multivues naïves, correspondant à des instances de l'algorithme MVSim, à savoir Cosinus MV, χ-Sim+Moyenne et χ-Sim+CSPA, qui nous fournissent ainsi une performance multivue de base. Nous remarquons que les résultats de MVSim sont meilleurs sur les 8 jeux de données multivues, par rapport à ces trois méthodes. La méthode χ-Sim+Moyenne obtient d'ailleurs systématiquement un meilleur résultat que Cosinus MV, ce qui n'est pas vraiment surprenant, étant donné que les performances monovues de l'algorithme χ-Sim sont également systématiquement meilleures que celles du Cosinus.

La méthode que nous avons nommée χ-Sim+CSPA nous fournit une comparaison simple avec une méthode de consensus de classification. Les résultats qu'elle obtient sont systématiquement inférieurs à ceux de l'algorithme MVSim, ce qui démontre bien l'apport de notre

méthode dans le domaine, par rapport aux approches de consensus. On remarque également qu'elle obtient systématiquement un moins bon résultat que χ-Sim+Moyenne, alors que les paramètres testés sont exactement les mêmes, ce qui signifie que cette fonction d'agrégation élémentaire n'est peut-être pas pertinente dans notre étude. Pour compléter cette analyse, on remarque également qu'elle ne parvient à obtenir un meilleur score que la meilleur méthode monovue, pour 2 jeux de données (sur 8).

De plus, nous remarquons que, pour les 8 jeux de données multivues dont nous disposons, les performances de l'algorithme MVSim sont supérieures à l'algorithme MVSC. Nuançons simplement ce résultat concernant le jeu de données ReutersEN, où le gain en précision micro-moyennée est de 0,001, ce qui représente environ 1 seul document.

Comme nous l'évoquions précédemment, nous ne disposons pas d'implémentation de l'algorithme MVKM, et nous ne pouvons que citer sa performance sur le jeu de données CiteSeer, que Drost *et al.* (2006) utilisent pour le tester. La Tab. 11 fournit ainsi les résultats des différentes approches sur le jeu CiteSeer, mais en terme d'entropie (H), car c'est la mesure privilégiée par les auteurs. Rappelons alors que contrairement à la précision micro-moyennée, une valeur d'entropie plus faible indique une meilleure performance.

Données	Meilleur monovue	Cosinus MV	χ-Sim +Moyenne	MVKM	MVSim
CiteSeer	1,27	1,42	1,19	1,60	**1,07**

Table 11 – Résultats sur CiteSeer des méthodes multivues incluant MVKM, exprimés en entropie (H).

Lorsque l'on utilise l'entropie et que l'on inclue alors l'algorithme MVKM, c'est encore MVSim qui obtient le meilleur résultat. Il est notamment remarquable que même le meilleur résultat des méthodes monovues obtienne une meilleur performance que MVKM, il s'agit en l'occurrence de l'algorithme χ-Sim sur la vue V_2, représentant les co-occurrences des mots dans les articles.

En conclusion de cette section, on peut dire que l'algorithme MVSim obtient de très bons résultats, et que l'intérêt d'utiliser des fonctions d'agrégations et de réinitialiser les instances de χ-Sim apporte un véritable gain en performance.

IV-3.3 Sensibilité aux paramètres

Dans cette section, nous allons étudier la sensibilité de l'algorithme MVSim à différents paramètres :

- le nombre d'itérations T_{MV}, que nous allons faire varier entre 1 et 10 avec un pas de 1 ;
- la fonction d'agrégation élémentaire F, pour laquelle nous allons tester le Minimum, le Maximum, la Moyenne et la fonction CSPA ;
- le paramètre $\alpha(t)$, qui peut valoir soit λ^{t-1}, soit $\frac{\lambda^{t-1}}{1+\lambda^{t-1}}$;
- et le paramètre d'amortissement λ, que nous allons faire varier entre 0 et 1 avec un pas de 0,2.

Notons que nous n'étudierons pas la sensibilité de MVSim aux paramètres de l'algorithme χ-Sim déjà étudiés dans la Sec. IV-2.3, qui sont les paramètres k, p et T qui seront alors fixés à leur valeur par défaut, respectivement 0,8, 0,6 et 1.

Afin de valider la robustesse de l'algorithme vis-à-vis de ces paramètres, nous allons réaliser quatre graphes permettant chacun d'analyser les variations de performance en terme de précision micro-moyennée de l'algorithme MVSim. Afin de ne pas surcharger les graphes, nous ne tracerons les courbes que pour quelques jeux de données représentatifs, sachant que les variations sont semblables pour les autres. Chaque point sera donc la moyenne de la précision micro-moyennée pour un jeu de données, et les barres d'erreur représentent les écarts-types. La moyenne et l'écart-type pour chaque point est calculée à partir des différents échantillons du jeu s'il y en a, et aussi des différentes valeurs des autres paramètres.

La Fig. 25 représente la variation de précision micro-moyennée en fonction du nombre d'itérations T_{MV} de l'algorithme MVSim. Par rapport à la variation des performances de χ-Sim en fonction du nombre d'itérations T, l'augmentation de la précision micro-moyennée en fonction de T_{MV} pour MVSim est moins nette, et dépend des jeux de données. On voit par exemple que pour le jeu de données IMDb, il n'y a pas d'amélioration significative, alors que pour CiteSeer les performances de l'algorithme augmentent visiblement pour les premières itérations, puis se stabilisent. Rappelons également que nous avons comparer les performances de l'algorithme MVSim avec ce que nous avons appelé Cosinus MV (une version « naïve » de MVSim avec en particulier $T_{\text{MV}} = 1$) afin de valider l'apport des itérations. En effet, les résultats de Cosinus MV étaient significativement inférieurs à ceux de MVSim.

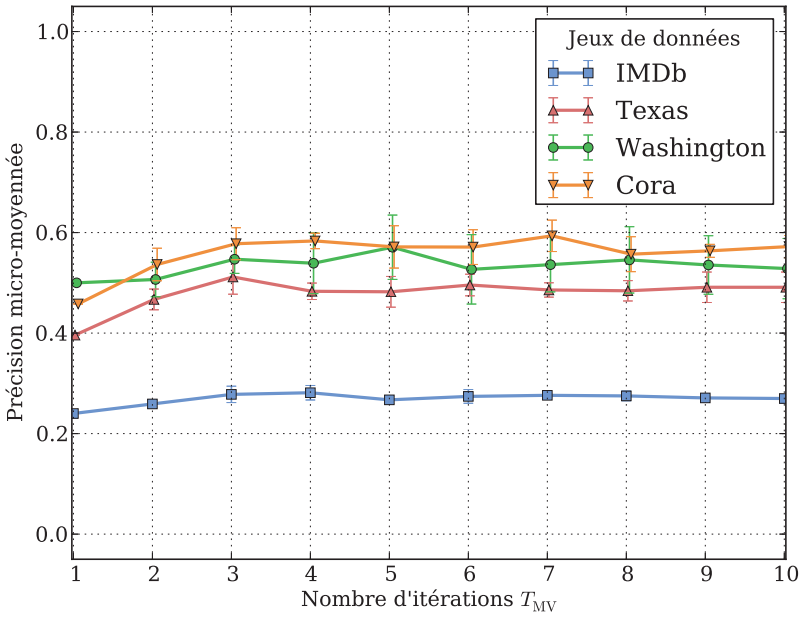


Figure 25 – Variations des performances de MVSim en fonction du nombre d'itérations T_{MV}. Les valeurs sont les moyennes (et écarts-types) des précisions micro-moyennées pour différentes valeurs des paramètres restants.

La Fig. 26 représente la variation de précision micro-moyennée en fonction de la fonction d'agrégation élémentaire F de l'algorithme MVSim. On observe que le choix de cette fonction a un impact sur la performance de l'algorithme, avec de fortes variations suivant la fonction F choisie. En particulier pour les jeux de données Cora et CiteSeer, la précision moyenne observée varie respectivement de 0,30 à 0,53, et de 0,22 à 0,53. Malgré cette relative sensibilité, on remarque que la fonction Moyenne entraîne systématiquement la meilleure ou la seconde meilleure performance, et elle est par conséquent la stratégie choisie par défaut. La fonction Maximum obtient de bons résultats sur l'ensemble des jeux testés et elle est par conséquent un choix raisonnable également. Quant aux fonctions Minimum et CSPA, on remarque qu'elles obtiennent des résultats irréguliers sur les différents jeux de données, et sont donc à écarter dans un contexte non supervisé.

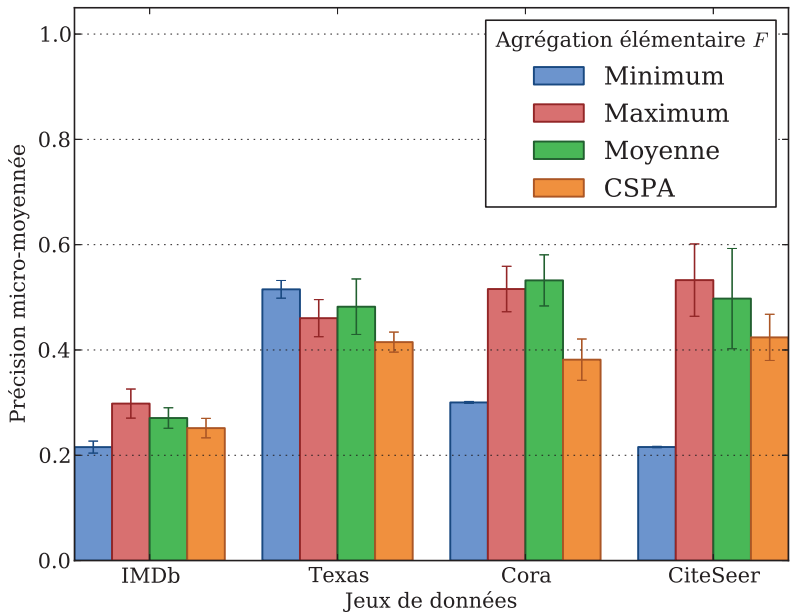


Figure 26 – Variations des performances de MVSim en fonction de la fonction d'agrégation élémentaire F. Les valeurs sont les moyennes (et écarts-types) des précisions micro-moyennées pour différentes valeurs des paramètres restants.

La Fig. 27 représente la variation de précision micro-moyennée en fonction du paramètre $\alpha(t)$ de l'algorithme MVSim. L'objectif était alors de voir si l'un des deux schémas proposés obtenait une meilleure performance, sachant que nous avons fait la moyenne pour toutes les valeurs de λ entre 0 et 1 avec un pas de 0,2. Bien que le schéma $\alpha(t) = \frac{\lambda^{t-1}}{1+\lambda^{t-1}}$ fournisse la meilleure précision micro-moyennée moyenne pour les quatre jeux de données utilisés, la différence par rapport au schéma $\alpha(t) = \lambda^{t-1}$ n'est pas significative au vu des valeurs des écarts-types. Par conséquent, les deux schémas sont viables et seront indifféremment utilisés.

La Fig. 28 représente la variation de précision micro-moyennée en fonction paramètre d'amortissement λ de l'algorithme MVSim. Rappelons que les valeurs sont les moyennes et écarts-types des précisions micro-moyennées, par rapport aux différentes fonctions d'agrégation élémentaire F et aux deux différents schémas possibles pour $\alpha(t)$. On observe alors que

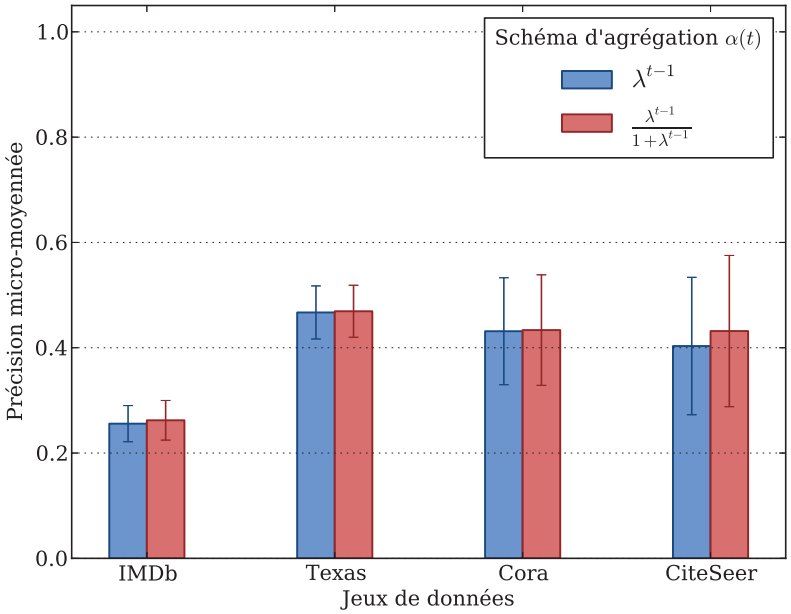


Figure 27 – Variations des performances de MVSim en fonction du schéma pour $\alpha(t)$. Les valeurs sont les moyennes (et écarts-types) des précisions micro-moyennées pour différentes valeurs des paramètres restants.

la performance de l'algorithme est très stable vis-à-vis de ce paramètre, ce qui démontre une robustesse satisfaisante de la méthode. Il semblerait cependant que les performances soient meilleures pour des valeurs élevées de λ et la valeur que nous choisirons par défaut sera 0,6.

En conclusion de cette étude de la sensibilité de l'algorithme MVSim à ces paramètres T_{MV}, F, $\alpha(t)$ et λ, nous pouvons affirmer que ses performances sont stables pour différentes valeurs de ces paramètres, et qu'il est donc relativement facile de l'appliquer à des tâches d'apprentissage non supervisé pour lesquelles il est délicat de choisir les paramètres.

IV-4 Application de MVSim au traitement d'une matrice de grande taille

Nous avons proposé dans la Sec. III-6.2 d'utiliser l'algorithme MVSim pour traiter une matrice de grande taille, pour laquelle l'utilisation directe de l'algorithme χ-Sim ne serait pas possible, soit à cause d'un temps de calcul trop long, soit à cause du stockage des matrices de similarité, trop grandes pour la mémoire vive de l'ordinateur exécutant l'algorithme. Nous avons alors proposé de découper cette matrice, soit uniquement ses colonnes, soit à la fois ses colonnes et ses lignes, et nous cherchons alors à valider l'intérêt de telles approches dans cette section.

Rappelons également que nous allons utiliser pour ces expérimentations la famille de jeux de données présentée dans la Sec. IV-1.3, nous fournissant un ensemble de matrices de co-occurrences documents - mots issues de la collection classique NG20. Grâce à ces jeux de

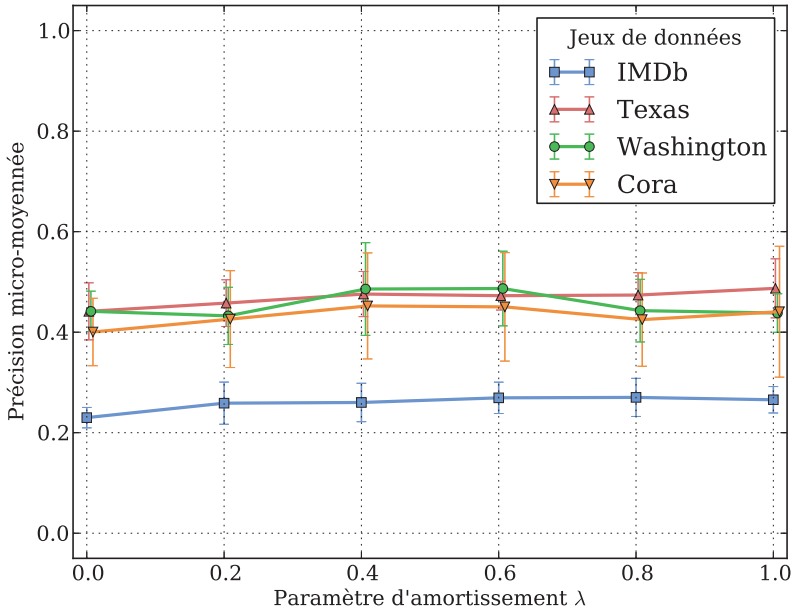


FIGURE 28 – Variations des performances de MVSim en fonction paramètre d'amortissement λ. Les valeurs sont les moyennes (et écarts-types) des précisions micro-moyennées pour différentes valeurs des paramètres restants.

données de taille différente, nous allons pouvoir faire varier le nombre de sous-ensembles, le nombre total de documents et de mots, et observer les variations de performance qui en résultent.

IV-4.1 Découpage selon une seule dimension

Dans cette section, nous allons appliquer le protocole que nous avons proposé dans la Sec. III-6.2.a, afin de découper l'ensemble des colonnes d'une matrice de grande taille $\mathbf{R}$ en M sous-ensembles. Nous avions alors calculé le gain en temps de calcul et l'espace mémoire obtenu en utilisant une telle approche, par rapport à l'utilisation directe de χ-Sim sur $\mathbf{R}$, et nous allons maintenant essayer de répondre, grâce à des expérimentations, aux deux questions suivantes :

- pour un nombre total de mots fixé, est-ce qu'augmenter le nombre de sous-ensembles de mots, dégrade significativement les performances de classification ?
- à temps de calcul et à mémoire fixés, peut-on améliorer la qualité de la classification en augmentant le nombre total de mots (en ajoutant des sous-ensembles de mots) ?

L'ensemble des mots d'une matrice doit être divisé en plusieurs sous-ensembles, cette division est faite aléatoirement, car nous nous plaçons dans un contexte d'apprentissage non supervisé, et ne disposons donc pas d'information *a priori* sur les mots. Les paramètres de MVSim ne sont pas optimisés dans cette section, et sont simplement fixés à des valeurs par défaut : $T_{\text{MV}} = 5$, $\lambda = 0{,}8$, $k = 0{,}8$, $p = 0{,}2$, car l'objectif est simplement d'observer les variations de performance de l'algorithme.

Réduire le temps de calcul et l'espace mémoire

Afin de répondre à la première question, nous fixons le nombre total de mots à 4000, et nous augmentons progressivement le nombre de sous-ensembles, i.e. le nombre de mots utilisés par chaque instance de χ-Sim diminue en conséquence : 1 division de 4000 mots, 2 sous-ensembles de 2000 mots, etc. Dans cette configuration, la quantité totale d'information utilisée par MVSim reste la même, mais le temps d'exécution diminue quand le nombre de sous-ensembles augmente. Cependant, l'information étant de moins en moins centralisée, on s'attend à une dégradation de la qualité de la classification des documents. Nous répétons de plus cette expérience pour différents nombre de documents. La Fig. 29 présente l'évolution des moyennes (et écarts-types) des précisions micro-moyennées obtenues par MVSim sur les 10 échantillons, pour les différents jeux de données (différents nombre de documents), en fonction du nombre de sous-ensembles de mots.

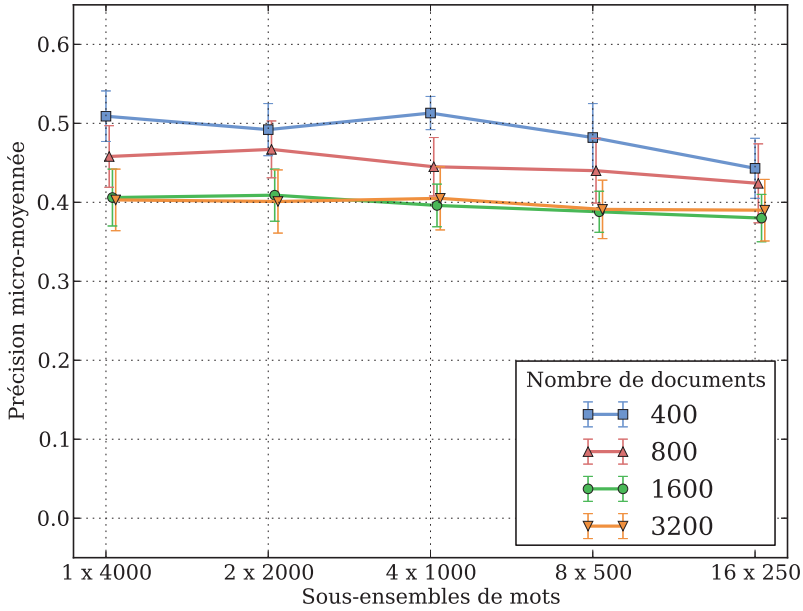


Figure 29 – Performance de MVSim pour différents nombres de sous-ensembles de mots, avec un nombre total de mots fixés. Les valeurs sont les moyennes (et écarts-types) des précisions micro-moyennées sur les 10 échantillons.

La Fig. 29 nous permet de confirmer la tendance à la dégradation des résultats lorsque le nombre de sous-ensembles augmente. Cependant, cette dégradation n'est pas très importante : par exemple si l'on compare les résultats pour 1 et pour 8 sous-ensembles, on constate une baisse relative de la précision micro-moyennée de 5%, mais alors que le temps d'exécution a été 64 fois plus rapide, et que l'espace mémoire nécessaire 8 fois moindre. De plus, on remarque que c'est pour le plus petit jeu de données (400 documents) que la dégradation est la plus importante lorsque le nombre de sous-ensembles augmente. On peut expliquer cette forte dégradation par le fait que le jeu étant déjà de faible taille, le fait de trop diviser les mots induit une grande perte d'information, et cette approche n'est pas intéressante pour un jeu de données de si petite taille.

En conclusion, lorsque la quantité totale d'information – e.g. nombre de mots – est limitée, il y a clairement un compromis entre la rapidité d'exécution (plus de sous-ensembles impliquant un temps d'exécution plus court), et la qualité de la classification (moins de sous-ensembles impliquant une meilleure qualité). Si l'on dispose des ressources suffisantes pour ne pas avoir à diviser la matrice de données, il est bien entendu plus intéressant de ne pas le faire. Mais même dans ce cas, si l'on veut par exemple tester de nombreuses valeurs différentes pour les paramètres de l'algorithme, avoir un temps d'exécution réduit est un atout indéniable.

Améliorer la qualité de la classification

Maintenant, afin de répondre à la seconde question, nous ne supposons plus que le nombre total de mots est constant, mais que le nombre de mots par sous-ensemble l'est. Ainsi dans ce contexte, nous supposons qu'un nœud de calcul ne peut pas traiter plus d'un certain nombre de mots, et que la seule façon d'utiliser plus d'information est de considérer plusieurs sous-ensembles. Nous fixons donc le nombre de mots par instance de χ-Sim à 500, et ajoutons plus de sous-ensembles, passant de 1 sous-ensemble à 2 puis 4 et finalement 8, atteignant finalement 4000 mots au total. On augmente ainsi le nombre total de mots considérés par l'algorithme MVSim, mais il est primordial de rappeler que le temps d'exécution de l'algorithme est lui constant, car on suppose que chaque instance de χ-Sim est exécutée par une unité de calcul distincte. La Fig. 30 présente l'évolution des moyennes (et des écarts-types) des précisions micro-moyennées sur les 10 échantillons, pour les différents jeux de données (différents nombre de documents), en fonction du nombre de sous-ensembles de mots, et donc du nombre total de mots considérés.

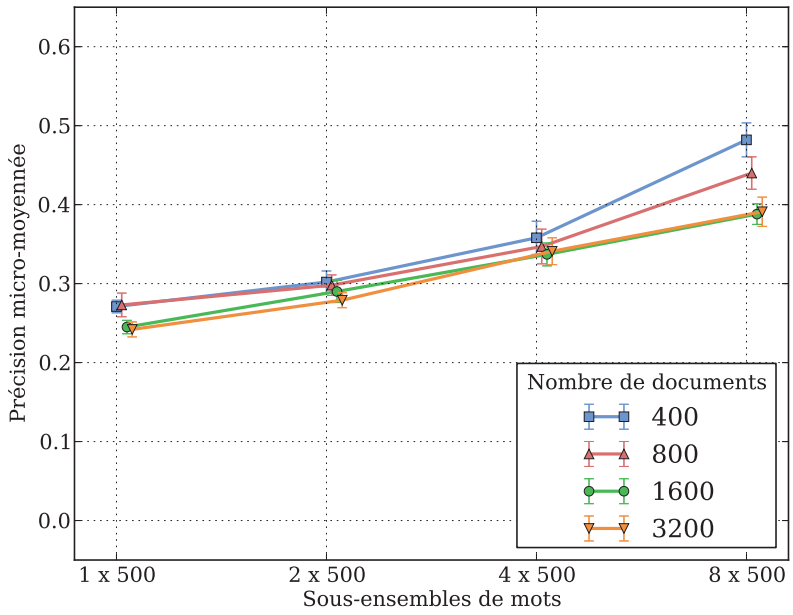


Figure 30 – Performance de MVSim pour différents nombres de sous-ensembles de mots, avec un nombre de mots par sous-ensemble fixé. Les valeurs sont les moyennes (et écarts-types) des précisions micro-moyennées sur les 10 échantillons.

L'observation de la Fig. 30, confirme qu'il est possible de nettement améliorer la qualité de la classification, à temps de calcul constant, en utilisant plus de mots. En effet, en passant de 500 mots à 4000 mots considérés au total par l'algorithme MVSim, et malgré le fait que ces 4000 mots soient séparés en 16 sous-ensembles, le gain relatif moyen en terme de précision micro-moyennée que l'on observe est de 65%. Pour être complet, il faut ajouter qu'avec les tailles des matrices utilisées pour ces expérimentations, il est possible de ne pas diviser les 4000 mots, ce qui résulte en une meilleure classification bien entendu. Ces résultats sont ceux correspondant à 1 division dans sur la Fig. 29.

En conclusion, on peut confirmer que l'application de MVSim pour le découpage de matrices de grande taille est réellement intéressant dans ce contexte (on ne divise que les colonnes). Ainsi, l'utilisateur est moins contraint à se limiter à un nombre de mots restreint qui entraverait la performance de l'algorithme χ-Sim. Comme dans le cas monovue cependant, on note que cette performance n'augmente pas nécessairement de façon linéaire avec le nombre de mots, et qu'il convient de fixer *a priori* un nombre raisonnable de mots – en se basant sur le nombre de documents de la collection par exemple – afin d'éviter d'ajouter des mots non pertinents.

IV-4.2 Découpage selon les deux dimensions

Dans cette section, nous allons appliquer le protocole décrit dans la Sec. III-6.2.b, afin de découper les deux dimensions (lignes et colonnes) d'une matrice de grande taille $\mathbf{R}$ en M^2 sous-matrices. Nous avions alors calculé le gain en temps de calcul et l'espace mémoire qu'il est possible de gagner en utilisant une telle approche, par rapport à l'utilisation direct de χ-Sim sur $\mathbf{R}$, et nous allons maintenant évaluer les variations sur la qualité de la classification résultante qu'un tel découpage implique.

A l'instar de la section précédente, les sous-ensembles de documents et de mots sont construits de façon aléatoire, car nous ne supposons aucune connaissance *a priori* sur les données, dans un contexte d'apprentissage non supervisé. De même, les paramètres de l'algorithme MVSim ne sont pas optimisés et sont simplement fixés à $T_{\text{MV}} = 10$, $\lambda = 0{,}9$, $k = 0{,}8$, $p = 0{,}8$ car l'objectif est simplement d'observer les variations de performance de l'algorithme.

Pour ces expérimentations, nous avons considérer deux jeux de données de l'ensemble décrits dans la Sec. IV-1.3, à savoir le jeu comportant 3200 documents et 2000 mots, et le jeu comportant 6400 documents et 4000 mots. Puis, à partir de ces deux jeux, nous avons fait varier le nombre de sous-ensembles de la façon suivante : 1×1, 2×4, 4×4, 8×8, 16×16, le premier nombre représentant le nombre de sous-ensembles de documents, et le second le nombre de sous-ensembles de mots. La Fig. 31 présente l'évolution des moyennes (et des écarts-types) des précisions micro-moyennées sur les 10 échantillons, pour les différents jeux de données, en fonction du nombre de sous-ensembles de documents et de mots.

A l'observation de la Fig. 31, on s'aperçoit immédiatement que contrairement au cas traité dans la section précédente, où l'on ne divisait que l'ensemble des mots, la performance se dégrade significativement. En effet, pour les deux jeux de données utilisés, la baisse relative

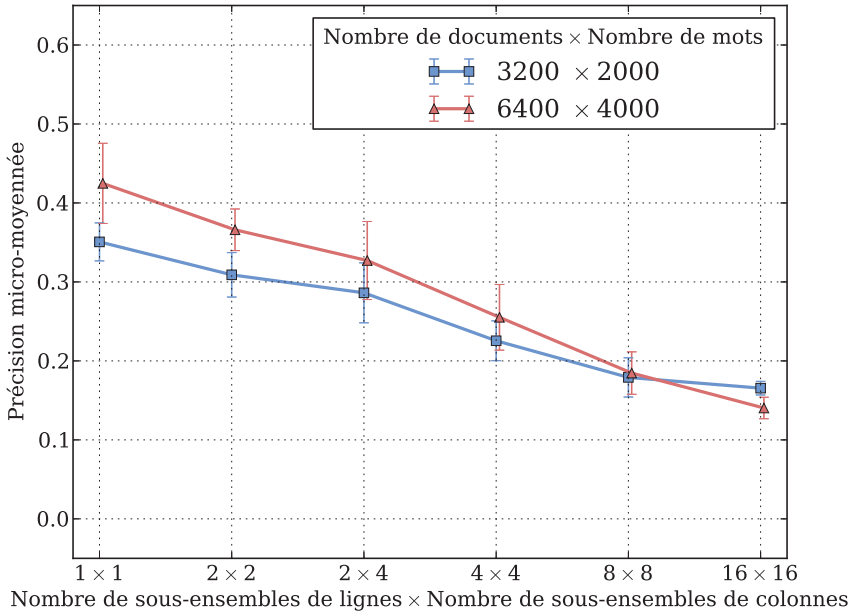


Figure 31 – Performance de MVSim pour différents nombres de sous-ensembles de documents et de mots. Les valeurs sont les moyennes (et écarts-types) des précisions micro-moyennées sur les 10 échantillons.

moyenne de précision micro-moyennée en passant de 1×1 à 16×16 est de 86%, tombant alors autour de 0,15. Rappelons au passage que le nombre de classes de ces jeux de données est 10, et qu'une classification aléatoire fournirait alors un score de 0,10 en moyenne.

Une interprétation de ce comportement est qu'avec cette approche, la matrice de similarité entre documents n'est que partiellement calculée par l'algorithme MVSim, et les valeurs de similarité entre des documents n'étant pas dans le même sous-ensemble sont ensuite approximées. C'est probablement cette étape d'approximation qui induit une forte dégradation des performances de classification, et il est possible que l'approche que nous avons proposée pour reconstruire ces similarités soit trop simple. Il faut cependant penser que l'objectif de cette approche est principalement de réduire le temps de calcul des similarités entre toutes les paires de documents décrit par une telle matrice, et la méthode pour approximer ces similarités manquantes ne soient alors pas être trop coûteuse en temps.

En conclusion de cette section, nous pouvons dire que cette approche ne semble pas encore prête à être utilisée pour des cas réels, car les performances sont trop rapidement dégradées en augmentant le nombre de sous-ensembles. Quoi qu'il en soit, rappelons que dans le cas où on l'utiliserait pour une tâche de classification, il faudrait alors calculer toutes les similarités manquantes, et nous avons vu que le gain en complexité temporelle serait nul. Cette approche ne doit cependant pas être totalement écartée car elle peut être utile pour d'autres tâches, pour lesquelles il ne serait pas nécessaire de connaître l'intégralité de la matrice de similarité.

IV-5 Conclusion du chapitre

Dans ce chapitre, nous avons réalisé de multiples expérimentations pour valider les performances des algorithmes χ-Sim et MVSim, respectivement proposés dans les Chap. II et III, qui permettent de calculer la matrice de similarité $\mathbf{S}_1$ à partir de laquelle on peut automatiquement classifier les instances de X_1. Au total, 15 jeux de données ont été impliqués dans ce chapitre, dont 8 jeux multivues. De plus, nous avons pu comparer nos deux algorithmes à des méthodes de l'état de l'art, nous permettant de valider l'apport des algorithmes χ-Sim et MVSim.

Les performances de l'algorithme χ-Sim pour lequel nous avons proposé plusieurs améliorations par rapport à la version originale (Bisson et Hussain 2008), sont très satisfaisantes, car il obtient soit le meilleur score (dans 60% des jeux de données), soit le second meilleur score (dans 40% des jeux de données).

Quant à l'algorithme MVSim que nous avons développé pour le calcul multivue de similarités, ils obtient également de bons résultats, à la fois par rapport aux méthodes monovues, validant ainsi l'intérêt de considérer les multiples vues d'un jeu de données ; et aussi par rapport aux méthodes multivues auxquelles nous avons comparé ses performances. De plus, nous avons réalisé des expérimentations pour le traitement d'une matrice de grande taille, pour laquelle on divise les colonnes, et éventuellement les lignes, en plusieurs sous-ensembles afin d'utiliser l'algorithme MVSim. Nous avons ainsi montré que ces approches peuvent en effet être intéressantes car, en particulier lorsqu'on ne divise que les colonnes (et que l'on cherche à classifier les objets décrits par les lignes), il est possible soit d'accélérer le calcul et de réduire la mémoire nécessaire, soit à temps de calcul équivalent, d'améliorer la qualité de la classification.

Conclusion

Résumé et contributions

Les mesures de similarités sont au cœur d'un grand nombre de tâches d'apprentissage automatique, et leur étude est particulièrement intéressante dans le cas d'objets décrits par un très grand nombre d'attributs. Dans cette thèse, nous nous sommes concentrés sur des tâches d'apprentissage non supervisé mais nos mesures sont extensibles à d'autres catégories de tâches en apprentissage supervisé ou semi-supervisé par exemple.

Dans le Chap. I, nous avons établi un panorama des différentes familles de méthodes utiles à la classification. Nous avons à la fois décrit différents modèles de représentation des données, mais aussi présenté les mesures classiques de distances et de similarités, ainsi que leur limites dans le cadre de données de grandes dimensions. Puis, nous avons présenté différentes familles de méthodes de classification à proprement parler, allant des méthodes classiques aux méthodes de co-classification qui classifient simultanément deux types d'objets, pour terminer par les méthodes de classification multivue qui traitent plusieurs types d'objets décrits par plusieurs relations, et donc plusieurs matrices. De plus, nous avons également décrit différentes approches de recherche de classification consensus pouvant s'appliquer à la fois aux données monovues classiques, et aux données multivues. Finalement, nous avons rappelé les différentes façons d'évaluer des classifications, en se concentrant sur le cas où l'on dispose pour ce faire des classes réelles des objets.

Dans le Chap. II, nous avons dans un premier temps présenté l'algorithme de calcul de co-similarité χ-Sim initialement mis au point par Bisson et Hussain (2008), afin de proposer différentes améliorations et études théoriques. Nous avons tout d'abord proposé un nouveau schéma de normalisation inspiré de la notion de mesure de similarité du Cosinus généralisé de Qamar et Gaussier (2009), et garantissant que la similarité entre un objet et lui-même soit toujours maximale. De plus, nous avons alors prouvé que les matrices de similarité ainsi calculées étaient semi-définies positives, permettant une meilleure interprétation théorique de l'approche. Nous avons ensuite introduit la notion de pseudo-norme en nous inspirant des travaux sur la norme L_k d'Aggarwal *et al.* (2001) et l'avons intégrée à la mesure χ-Sim. Par ailleurs, nous avons présenté une interprétation en terme de graphe de la mesure et ajouté une étape de seuillage des similarités les plus faibles en conséquence. Egalement d'après cette étude du graphe, nous avons proposé d'introduire un amortissement des co-occurrences d'ordres

supérieures dans la mesure. Finalement nous avons cherché à savoir si l'algorithme χ-Sim ne résolvait pas implicitement un problème d'optimisation.

Dans le Chap. III, nous avons proposé l'algorithme de calcul multivue de co-similarités MVSim, qui peut être vu comme une extension de χ-Sim aux cas des données multivues. En effet, l'algorithme MVSim calcule une matrice de similarité pour chacun des types d'objets présents dans les données, en utilisant toutes les vues des données, i.e. toutes les matrices de relations disponibles pour décrire ces objets. L'algorithme MVSim s'appuie alors sur autant d'instances de χ-Sim qu'il a de vues dans les données, et sur des fonctions d'agrégation Agg dont le rôle principal est de faire circuler l'information entre les différentes instances de χ-Sim et donc entre les différentes vues. Après avoir étudié la complexité de l'algorithme, nous avons détaillé deux applications possibles de MVSim à des données monovues, dans le cas où la matrice de relation est de trop grande dimension pour pouvoir directement y appliquer χ-Sim. Cette approche est alors basée sur le découpage des colonnes (et éventuellement des lignes) de cette matrice, créant ainsi un ensemble de sous-matrices qui seront alors utilisées par MVSim.

Finalement dans le Chap. IV, nous avons proposé différentes tâches de classification afin de comparer les performances des algorithmes χ-Sim et MVSim par rapport aux méthodes de l'état de l'art vues dans le Chap. I. Après avoir présenté un ensemble de jeux de données monovues et multivues, nous avons comparé les résultats de l'algorithme χ-Sim à des mesures de similarité ainsi qu'à des méthodes de co-classification de l'état de l'art, mais également à la version originale de l'algorithme. Nous avons ainsi pu validé les améliorations apportées à l'algorithme, et observé une performance moyenne de χ-Sim supérieure aux méthodes de l'état de l'art, obtenant systématiquement le meilleur ou le second meilleur score pour l'ensemble des jeux de données testés. De plus, nous avons réalisé une étude de sensibilité de χ-Sim à ses paramètres, et valider nos choix expérimentaux. Puis, nous avons comparé les résultats de l'algorithme MVSim aux résultats des méthodes monovues, et à des algorithmes de classification multivue, ainsi qu'à des versions « naïves » de l'algorithme, afin de valider ses performances. Premièrement, nous avons valider l'apport du multivue pour la majorité des jeux de données, les résultats de MVSim étaient meilleurs que ceux des méthodes monovues. De plus, les résultats obtenus par MVSim sont supérieurs à ceux des autres méthodes multivues que nous avons utilisées dans notre comparaison, et grâce aux versions « naïves » de l'algorithme, nous avons également pu valider différentes composantes de MVSim, telles que la fonction d'agrégation. A l'instar de ce que nous avions fait pour l'algorithme χ-Sim, nous avons étudié la sensibilité de MVSim à ses paramètres, et conclu à une robustesse satisfaisante de la méthode.

Egalement dans le Chap. IV, nous avons testé nos propositions d'appliquer l'algorithme MVSim à une matrice de grande dimension. Nous avons d'abord expérimenté le découpage des colonnes uniquement, et avons observé que le fait de découper la matrice n'impliquait qu'une faible dégradation de la qualité de la classification, pour un gain important en terme de complexité temporelle et spatiale. De plus, nous avons également montré qu'il est possible d'améliorer la classification en considérant plus d'attributs, pour un temps de calcul inchangé. Finalement, nous avons également expérimenté le découpage des lignes et des colonnes de la

matrice, impliquant alors une reconstruction des similarités manquantes avant de procéder à la classification. La dégradation observée lorsque l'on augmente le nombre de sous-matrices est alors plus importante que dans le cas précédent, mais permet toutefois de traiter des matrices qu'il ne serait pas possible d'étudier sans cela.

Limites et perspectives

Dans ce manuscrit, nous avons montré que les algorithmes χ-Sim pour les jeux de données monovues, et MVSim pour les jeux de données multivues, obtenaient de bons résultats en classification. Cependant, il convient de mentionner plusieurs limites à ces approches.

Malgré nos propositions de découper une matrice de grande dimension afin de pouvoir utiliser MVSim à partir de blocs de cette matrice, la complexité temporelle et surtout spatiale de algorithmes que nous avons proposés reste un point faible. En effet, le fait de devoir mémoriser les similarités entre toutes les paires d'objets auxquels on s'intéresse nécessite une taille de stockage très importante.

De plus, le temps de calcul pouvant également être relativement long, il serait intéressant de pouvoir mettre à jour les matrices de similarités sans avoir à tout recommencer si un nouveau document était ajouté au jeu de données. En effet, face à un scénario dit d'apprentissage en ligne, où les données ne sont pas figées, les algorithmes χ-Sim et MVSim ne peuvent pas apprendre de façon incrémentale de nouvelles similarités. Modifier ces algorithmes pour leur permettre de pouvoir mettre à jour des similarités face à de nouvelles données serait d'ailleurs une façon de les faire passer à l'échelle.

Par ailleurs, nous avons présenté une interprétation des itérations dans l'algorithme χ-Sim en terme de prise en compte des co-occurrences d'ordre supérieur. Cependant, il serait plus élégant et plus satisfaisant d'obtenir une convergence non triviale (différente de matrices de 1) de cette méthode. C'est dans ce but que nous avons fait une première proposition d'intégrer un amortissement des co-occurrences d'ordre supérieur, ce qui revient à pénaliser les chemins dans le graphe des documents en fonction de leur longueur. Bien que cette perspective semble prometteuse, nous ne sommes jusqu'à présent pas parvenus à obtenir de résultats expérimentaux satisfaisants à partir de cette approche. Ce travail n'a cependant pas été vain car nous avons intégré cette notion d'amortissement dans l'algorithme multivue MVSim.

Une piste de recherche intéressante serait également de modifier l'algorithme MVSim pour l'adapter à des tâches d'apprentissage supervisé ou semi-supervisé. De manière plus générale, on peut imaginer rendre l'architecture plus générale et remplacer l'algorithme χ-Sim par des instances d'autres méthodes, adaptées à ces différentes tâches. Par ailleurs, que ce soit pour ces tâches ou pour une tâche d'apprentissage non supervisé comme nous l'avons fait dans cette thèse, une amélioration intéressante de l'algorithme serait d'apprendre à pondérer les différentes vues. C'est alors surtout pour des tâches d'apprentissage non supervisé qu'il serait délicat de trouver un bon critère permettant d'apprendre des poids pertinents. Comme nous l'avons vu dans nos expérimentations de classification multivue avec MVSim, pour les jeux de données où l'algorithme n'obtient pas le meilleur résultat, c'est au détriment de

méthodes monovues, car ces jeux comportent des vues de qualité très inégale. Dans de tels cas, apprendre à les pondérer pour détecter qu'une vue n'est pas pertinente par exemple, permettrait certainement d'améliorer les performances de l'approche.

Concernant l'approche par découpage de matrices de grande dimension, le fait de constituer les sous-ensembles d'objets de façon parfaitement aléatoire est certainement une méthode trop naïve. Une perspective intéressante serait alors la mise au point d'une heuristique simple, i.e. n'ayant pas une complexité trop importante, permettant de créer des sous-ensembles d'objets similaires. Le fait de ne pas calculer les similarités des paires d'objets n'étant pas dans le même sous-ensemble serait alors moins problématique.

Les deux algorithmes que nous avons proposés dans cette thèse comportent chacun plusieurs paramètres, qu'il convient de fixer *a priori* dans un contexte d'apprentissage non supervisé. Dans le Chap. IV nous avons étudié la sensibilité de nos algorithmes à leurs paramètres, et conclu à une relative robustesse de ceux-ci. Cependant il faut aussi remarquer que lors des tests de classification, nous avons alors fait varier les paramètres, et présenté le meilleur résultat obtenu, ce qui représente une sélection de paramètres *a posteriori*. Bien que cette méthodologie de test ne nous paraissent pas en adéquation avec une tâche d'apprentissage non supervisé dans le sens qu'un utilisateur de telle méthode n'aurait pas la chance de pouvoir faire ce choix, il se trouve que c'est la méthodologie systématiquement utilisée dans les publications. Afin de palier ce problème, nous avions mené une étude préliminaire afin de déterminer automatiquement les paramètres k et p de l'algorithme χ-Sim, sans recourir aux classes réelles des objets. Grâce à des heuristiques relativement simples, nous parvenions à trouver des valeurs pour ces paramètres permettant de s'approcher presque toujours du maximum de performance de la méthode.

Toujours en lien avec le fait que ces méthodes d'apprentissage non supervisé comportent des paramètres, une perspective intéressante serait de systématiquement utiliser des méthodes de consensus de classification, à partir des multiples classifications construites pour des paramètres différents.

Bibliographie

Acar, E. et Yener, B. (2009). Unsupervised multiway data analysis : A literature survey. *IEEE Transactions on Knowledge and Data Engineering*, 21(1):6–20. 10

Aggarwal, C., Hinneburg, A. et Keim, D. (2001). On the surprising behavior of distance metrics in high dimensional space. *In* Bussche, J. et Vianu, V., éditeurs : *Database Theory — ICDT 2001*, volume 1973 de *Lecture Notes in Computer Science*, chapitre 27, pages 420–434. Springer Berlin Heidelberg, Berlin, Heidelberg. 15, 51, 89, 105

Albanese, D., Visintainer, R., Merler, S., SamanthaRiccadonna, Jurman, G. et Furlanello, C. (2012). mlpy : Machine learning python. 86, 93

Amini, M., Usunier, N., Goutte, C. *et al.* (2010). Learning from multiple partially observed views-an application to multilingual text categorization. *In Proceedings of the 23rd Annual Conference on Neural Information Processing Systems*, volume 1 de *NIPS'2010*, pages 28–36. 83

Arora, R., Gupta, M., Kapila, A. et Fazel, M. (2011). Clustering by left-stochastic matrix factorization. *In Proceedings of the 28th International Conference on Machine Learning*, ICML'11, pages 761–768, New York, NY, USA. ACM. 22

Banerjee, A., Basu, S. et Merugu, S. (2007). Multi-way clustering on relation graphs. *In Proceedings of the 2007 SIAM International Conference on Data Mining*, SIAM ICDM'2007. 10

Banerjee, A. et Ghosh, J. (2002). Frequency sensitive competitive learning for clustering on high-dimensional hyperspheres. *In Proceedings of the 2002 International Joint Conference on Neural Networks*, volume 2 de *IJCNN'2002*, pages 1590–1595. IEEE. 37

Bekkerman, R., El-Yaniv, R. et McCallum, A. (2005). Multi-way distributional clustering via pairwise interactions. *In Proceedings of the 22nd international conference on Machine learning*, ICML'2005, pages 41–48. ACM. 24, 25, 26

Bellman, R. et Kalaba, R. (1959). On adaptive control processes. *Automatic Control, IRE Transactions on*, 4(2):1–9. 15

Berkhin, P. (2002). Survey of clustering data mining techniques. Rapport technique, Accrue Software, San Jose, CA. 12, 15

Bickel, S. et Scheffer, T. (2005). Estimation of mixture models using co-em. *In Proceedings of the 22nd international conference on Machine learning*, pages 35–46. Springer. 24, 28, 37

Bisson, G. et Hussain, F. (2008). X-sim : A new similarity measure for the co-clustering task. *In Proceedings of the 7th International Conference on Machine Learning and Applications*, ICMLA'2008, pages 211–217. xv, 4, 41, 42, 45, 46, 47, 51, 54, 58, 79, 85, 104, 105

Blaschko, M. et Lampert, C. (2008). Correlational spectral clustering. *In Proceedings of the 2008 Computer Vision and Pattern Recognition conference*, CVPR'2008, pages 1–8. IEEE. 28

Blum, A. et Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. *In Proceedings of the eleventh annual conference on Computational learning theory*, COLT'1998, pages 92–100. ACM. 23, 27

Brand, M. (2006). Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications*, 415(1):20 – 30. 19

Chakraborti, S., Lothian, R., Wiratunga, N. et Watt, S. (2006). Sprinkling : supervised latent semantic indexing. *Advances in Information Retrieval*, pages 510–514. 18

Chaudhuri, K., Kakade, S., Livescu, K. et Sridharan, K. (2009). Multi-view clustering via canonical correlation analysis. *In Proceedings of the 26nd international conference on Machine learning*, ICML'2009, pages 129–136. 28

Chen, S., Wang, F. et Zhang, C. (2009). Simultaneous heterogeneous data clustering based on higher order relationships. *In Proceedings of the Workshop on Mining Graphs and Complex Structures*, MGCS'2007. 24

Cheng, Y. et Church, G. M. (2000). Biclustering of expression data. *In Proceedings of the 8th International Conference on Intelligent System for Molecular Biology*, ISMB'2010, pages 93–103. 18

de Carvalho, F., Lechevallier, Y. et de Melo, F. M. (2012). Partitioning hard clustering algorithms based on multiple dissimilarity matrices. *Pattern Recognition*, 45:447 – 464. 28

de Sa, V. R. (2005). Spectral clustering with two views. *In Proceedings of Workshop of Learning with Multiple Views*, ICML'2005. 27

Deerwester, S., Dumais, S. T., Furnas, G. W., Thomas et Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407. 18, 85

Dempster, A., Laird, N., Rubin, D. *et al.* (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38. 16

Dhillon, I. S. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. *In Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, SIGKDD'2001, pages 269–274. 18, 19, 20, 22, 82, 85, 86

Dhillon, I. S., Mallela, S. et Modha, D. S. (2003). Information-theoretic co-clustering. *In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, SIGKDD'2003, pages 89–98. 19, 36, 80

Dice, L. (1945). Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302. 14

Dimitriadou, E., Weingessel, A. et Hornik, K. (2001). Voting-merging : An ensemble method for clustering. *In Proceedings of the 2001 International Conference on Artificial Neural Networks*, ICANN'2001, pages 217–224. Springer. 34

Ding, C., He, X. et Simon, H. (2005). On the equivalence of nonnegative matrix factorization and spectral clustering. *In Proceedings of the 2005 SIAM International Conference on Data Mining*, numéro 4 de SIAM ICDM'2005, pages 606–610. 22

Drost, I., Bickel, S. et Scheffer, T. (2006). Discovering communities in linked data by multi-view clustering. *From Data and Information Analysis to Knowledge Engineering*, pages 342–349. 28, 37, 92, 95

Dudoit, S. et Fridlyand, J. (2003). Bagging to improve the accuracy of a clustering procedure. *Bioinformatics*, 19(9):1090–1099. 34

Fern, X. et Brodley, C. (2003). Random projection for high dimensional data clustering : A cluster ensemble approach. *In Proceedings of the 20th International Conference on Machine Learning*, volume 20 de *ICML'2003*, page 186. 30

Fern, X. et Brodley, C. (2004). Solving cluster ensemble problems by bipartite graph partitioning. *In Proceedings of the 21st International Conference on Machine learning*, ICML'2004, page 36. ACM. 31, 32, 33

Fred, A. et Jain, A. (2002). Data clustering using evidence accumulation. *In Pattern Recognition 2002*, volume 4, pages 276–280. 30

Friedman, N., Mosenzon, O., Slonim, N. et Tishby, N. (2001). Multivariate information bottleneck. *In Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 152–161. Morgan Kaufmann Publishers Inc. 24, 25

Frigui, H., Hwang, C. et Rhee, F. C.-H. (2007). Clustering and aggregation of relational data with applications to image database categorization. *Pattern Recognition*, 40(11):3053 – 3068. 24, 25, 28

Gower, J. et Legendre, P. (1986). Metric and euclidean properties of dissimilarity coefficients. *Journal of classification*, 3(1):5–48. 14

Greene, D. et Cunningham, P. (2009). A matrix factorization approach for integrating multiple data views. *Machine Learning and Knowledge Discovery in Databases*, pages 423–438. 22

Grimal, C. et Bisson, G. (2010). Classification à partir d'une collection de matrices. *In Conférence Francophone sur l'Apprentissage Automatique*, CAp'2010, pages 309–312. 65

Hathaway, R., Davenport, J. et Bezdek, J. (1989). Relational duals of the c-means clustering algorithms. *Pattern recognition*, 22(2):205–212. 25

Hofmann, T. (2001). Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1):177–196. 19

Hubert, L. et Arabie, P. (1985). Comparing partitions. *Journal of classification*, 2(1):193–218. 38

Iam-on, N., Boongoen, T. et Garrett, S. (2010). LCE : a link-based cluster ensemble method for improved gene expression data analysis. *Bioinformatics*, 26(12):1513–1519. 18

Ingaramo, D., Errecalde, M. et Rosso, P. (2010). A general bio-inspired method to improve the short-text clustering task. *In* Gelbukh, A., éditeur : *Computational Linguistics and Intelligent Text Processing*, volume 6008 de *Lecture Notes in Computer Science*, pages 661–672. Springer Berlin / Heidelberg. 18

Jaccard, P. (1901). *Bulletin de la Société vaudoise des sciences naturelles*. Numéro vol. 37. Impr. F. Blanchard. 14

Jain, A. K., Murphy, M. N. et Flynn, P. J. (1999). Data clustering : a review. *ACM computing surveys (CSUR)*, 31:264–323. 12, 15

Jones, E., Oliphant, T., Peterson, P. *et al.* (2001). SciPy : Open source scientific tools for Python. 85, 93

Karypis, G., Aggarwal, R., Kumar, V. et Shekhar, S. (1999). Multilevel hypergraph partitioning : applications in vlsi domain. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 7(1):69–79. 32

Karypis, G. et Kumar, V. (1999). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359. 31, 32

Kaufman, L. et Rousseeuw, P. (1990). *Finding Groups in Data : An Introduction to Cluster Analysis*. John Wiley John Wiley & Sons. 16, 25

Kontostathis, A. et Pottenger, W. (2006). A framework for understanding latent semantic indexing (lsi) performance. *Information Processing & Management*, 42(1):56–73. 19

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97. 36

Kumar, A. et Daume III, H. (2011). A co-training approach for multi-view spectral clustering. *In Proceedings of the 28th International Conference on Machine Learning*, ICML'2011, pages 393–400. 27, 83, 92, 93

Kummamuru, K., Dhawale, A. et Krishnapuram, R. (2003). Fuzzy co-clustering of documents and keywords. *In Proceedings of the 12th IEEE International Conference on Fuzzy Systems*, volume 2 de *FUZZ'2003*, pages 772–777. IEEE. 20, 21

Lemaire, B. et Denhière, G. (2008). Effects of high-order co-occurrences on word semantic similarities. *Current Psychology Letters - Behaviour, Brain and Cognition*, 18(1). 47

Li, T. et Ding, C. (2006). The relationships among various nonnegative matrix factorization methods for clustering. *In Proceedings of the 6th IEEE International Conference on Data Mining*, ICDM'2006, pages 362–371. IEEE. 22

Li, T. et Ding, C. (2008). Weighted consensus clustering. *In Proceedings of the 8th SIAM International Conference on Data Mining*, volume 1 de *SIAM ICDM'2008*, pages 798–809, Atlanta, USA. 34

Liu, N., Zhang, B., Yan, J., Yang, Q., Yan, S., Chen, Z., Bai, F. et Ma, W. (2004). Learning similarity measures in non-orthogonal space. *In Proceedings of the thirteenth ACM international conference on Information and knowledge management*, CIKM'2004, pages 334–341. ACM. 18, 42, 85, 86

Long, B., Wu, X., Zhang, Z. et Yu, P. (2006a). Unsupervised learning on k-partite graphs. *In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, SIGKDD'2006, pages 317–326, New York, NY, USA. ACM. 18, 27, 28, 80

Long, B., Zhang, Z., Wú, X. et Yu, P. S. (2006b). Spectral clustering for multi-type relational data. *In Proceedings of the 23rd International Conference on Machine Learning*, ICML'2006, pages 585–592. ACM. 24

Long, B., Zhang, Z. et Yu, P. (2005). Co-clustering by block value decomposition. *In Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, SIGKDD'2005, pages 635–640. ACM. 36

Madeira, S. et Oliveira, A. (2004). Biclustering algorithms for biological data analysis : a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):24–45. 18

Monti, S., Tamayo, P., Mesirov, J. et Golub, T. (2003). Consensus clustering : a resampling-based method for class discovery and visualization of gene expression microarray data. *Machine learning*, 52(1):91–118. 31

Ng, A., Jordan, M. et Weiss, Y. (2002). On spectral clustering : Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856. 27, 32

Ng, R. et Han, J. (2002). CLARANS : a method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*, 14(5):1003 – 1016. 16

Oh, C., Honda, K. et Ichihashi, H. (2001). Fuzzy clustering for categorical multivariate data. *In IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th*, volume 4, pages 2154–2159. IEEE. 20, 21

Pedrycz, W. (2002). Collaborative fuzzy clustering. *Pattern Recognition Letters*, 23(14):1675–1686. 24

Qamar, A. M. et Gaussier, E. (2009). Online and batch learning of generalized cosine similarities. *In Proceedings of the 9th IEEE International Conference on Data Mining*, ICDM'2009, pages 926–931, Washington, DC, USA. IEEE Computer Society. 45, 48, 49, 58, 105

Rege, M., Dong, M. et Fotouhi, F. (2008). Bipartite isoperimetric graph partitioning for data co-clustering. *Data Mining and Knowledge Discovery*, 16(3):276–312. 18, 22

Salton, G. (1983). *Introduction to modern information retrieval*. McGraw-Hill. 10

Slonim, N. et Tishby, N. (2000). Document clustering using word clusters via the information bottleneck method. *In Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR'2000, pages 208–215. ACM. 19

Speer, N., Spieth, C. et Zell, A. (2004). A memetic clustering algorithm for the functional partition of genes based on the gene ontology. *In Proceedings of the 2004 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, CIBCB'04, pages 252–259. IEEE. 18

Strehl, A. et Ghosh, J. (2003). Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *The Journal of Machine Learning Research*, 3:583–617. 29, 31, 32, 33, 37

Tang, W., Lu, Z. et Dhillon, I. S. (2009). Clustering with multiple graphs. *In Proceedings of the 2009 edition of the IEEE International Conference on Data Mining*, ICDM'09, pages 1016–1021. 22, 24, 27

Taskar, B., Segal, E. et Koller, D. (2001). Probabilistic classification and clustering in relational data. *In Proceedings of the 17th International Joint Conference on Artificial Intelligence*, IJCAI'2001, pages 870–878. 26

Tishby, N., Pereira, F. C. et Bialek, W. (1999). The information bottleneck method. *Proceedings of 37th Annual Allerton Conference on Communication Control and Computing*, pages 368–377. 25

Tjhi, W.-C. et Chen, L. (2005). Fuzzy co-clustering of web documents. *In International Conference on Cyberworlds*, pages 7 pp. –551. 20, 21

Topchy, A., Jain, A. et Punch, W. (2003). Combining multiple weak clusterings. *In Proceedings of the 3rd IEEE International Conference on Data Mining*, ICDM'2003, pages 331–338. IEEE. 33, 34

Topchy, A., Jain, A. et Punch, W. (2005). Clustering ensembles : Models of consensus and weak partitions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1866–1881. 33, 34

Wakabayashi, Y. (1998). The complexity of computing medians of relations. 29

Wang, J., Zeng, H., Chen, Z., Lu, H., Tao, L. et Ma, W. (2003). Recom : reinforcement clustering of multi-type interrelated data objects. *In Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR'2003, pages 274–281. ACM. 24, 26

Wang, X., Ma, W., Xue, G. et Li, X. (2004). Multi-model similarity propagation and its application for web image retrieval. *In Proceedings of the 12th annual ACM international conference on Multimedia*, pages 944–951. ACM. 42

Wang, X., Sun, J., Chen, Z. et Zhai, C. (2006). Latent semantic analysis for multiple-type interrelated data objects. *In Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR'2006, pages 236–243, New York, NY, USA. ACM. 24

Ward, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244. 17

Wiswedel, B. et Berthold, M. (2007). Fuzzy clustering in parallel universes. *International journal of approximate reasoning*, 45(3):439–454. 23, 24

Wiswedel, B., Höppner, F. et Berthold, M. (2010). Learning in parallel universes. *Data Mining and Knowledge Discovery*, 21(1):130–152. 23, 24

Xu, R., Wunsch, D. *et al.* (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678. 12, 15, 17

Yen, L., Fouss, F., Decaestecker, C., Francq, P. et Saerens, M. (2009). Graph nodes clustering with the sigmoid commute-time kernel : A comparative study. *Data Knowledge & Engineering*, 68(3):338–361. 18, 22, 85

Zanghi, H., Volant, S. et Ambroise, C. (2010). Clustering based on random graph model embedding vertex features. *Pattern Recognition Letters*, 31(9):830 – 836. 18, 22

Zeng, H., Chen, Z. et Ma, W. (2002). A unified framework for clustering heterogeneous web objects. *In Proceedings of the Third International Conference on Web Information Systems Engineering*, WISE'2002, pages 161–170. IEEE. 24, 26

Zhou, D., Orshanskiy, S., Zha, H. et Giles, C. (2007). Co-ranking authors and documents in a heterogeneous network. *In Proceedings of the 7th IEEE International Conference on Data Mining*, ICDM'2007, pages 739–744. IEEE. 27

Zipf, G. (1949). Human behavior and the principle of least effort. 11

Annexe A

Preuves

Proposition 3. *La fonction J définie dans le problème d'optimisation $(\mathcal{P})$ est convexe, i.e. :*

$$\forall \mathbf{s},\mathbf{s}' \in \mathbb{R}^{\eta} \text{ et } \forall t \in [0,1],\ J(t\,\mathbf{s} + (1-t)\,\mathbf{s}') \leqslant t\,J(\mathbf{s}) + (1-t)\,J(\mathbf{s}') \tag{24}$$

Démonstration. Nous introduisons une nouvelle notation $\hat{\mathbf{S}}_1 = \mathbf{S}_1 - \mathbf{R}\,\mathbf{S}_2\,\mathbf{R}^\top$ de terme général $\hat{s}^1_{ij}$ et $\hat{\mathbf{S}}_2 = \mathbf{S}_2 - \mathbf{R}^\top\,\mathbf{S}_1\,\mathbf{R}$ de terme général $\hat{s}^2_{ij}$. Pour plus de clarté, nous introduisons également $\sum_{ij}$ pour désigner $\sum_{i=1}^{\eta_1}\sum_{j=1}^{\eta_1}$ et $\sum_{kl}$ pour désigner $\sum_{k=1}^{\eta_2}\sum_{l=1}^{\eta_2}$. En utilisant cette nouvelle notation, nous ré-écrivons J comme suit :

$$\begin{aligned} J(\mathbf{s}) &= \sum_{i=1}^{\eta_1}\sum_{j=1}^{\eta_1}\left(s^1_{ij} - \sum_{k=1}^{\eta_2}\sum_{l=1}^{\eta_2} r_{ik}\,r_{jl}\,s^2_{kl}\right)^2 + \sum_{k=1}^{\eta_2}\sum_{l=1}^{\eta_2}\left(s^2_{kl} - \sum_{i=1}^{\eta_1}\sum_{j=1}^{\eta_1} r_{ik}\,r_{jl}\,s^1_{ij}\right)^2 \\ &= \sum_{ij}(\hat{s}^1_{ij})^2 \qquad\qquad + \sum_{kl}(\hat{s}^2_{kl})^2 \end{aligned}$$

$\forall \mathbf{s},\mathbf{s}' \in \mathbb{R}^{\eta}$ et $\forall t \in [0,1]$, on a :

$$\begin{aligned} J(t\,\mathbf{s} + (1-t)\,\mathbf{s}') = &\sum_{ij}\left[t\,s^1_{ij} + (1-t)\,s'^1_{ij} - \sum_{kl} r_{ik}\,r_{jl}\left(t\,s^2_{kl} + (1-t)\,s'^2_{kl}\right)\right]^2 \\ &+ \sum_{kl}\left[t\,s^2_{kl} + (1-t)\,s'^2_{kl} - \sum_{ij} r_{ik}\,r_{jl}\left(t\,s^1_{ij} + (1-t)\,s'^1_{ij}\right)\right]^2 \\ = &\sum_{ij}\left(t\,\hat{s}^1_{ij} + (1-t)\,\hat{s'}^1_{ij}\right)^2 + \sum_{kl}\left(t\,\hat{s}^2_{kl} + (1-t)\,\hat{s'}^2_{kl}\right)^2 \\ = &\ t^2\left(\sum_{ij}(\hat{s}^1_{ij})^2 + \sum_{kl}(\hat{s}^2_{kl})^2\right) \\ &+ (1-t)^2\left(\sum_{ij}(\hat{s'}^1_{ij})^2 + \sum_{kl}(\hat{s'}^2_{kl})^2\right) \\ &+ 2\,t\,(1-t)\left(\sum_{ij}\hat{s}^1_{ij}\,\hat{s'}^1_{ij} + \sum_{kl}\hat{s}^2_{kl}\,\hat{s'}^2_{kl}\right) \end{aligned}$$

et

$$t\,J(\mathbf{s}) + (1-t)\,J(\mathbf{s}') = t\left(\sum_{ij}(\hat{s}^1_{ij})^2 + \sum_{kl}(\hat{s}^2_{kl})^2\right) + (1-t)\left(\sum_{ij}(\hat{s'}^1_{ij})^2 + \sum_{kl}(\hat{s'}^2_{kl})^2\right)$$

Ainsi $J(t\,\mathbf{s} + (1-t)\,\mathbf{s}') \leqslant t\,J(\mathbf{s}) + (1-t)\,J(\mathbf{s}')$ est équivalent à :

$$\begin{aligned}
t\,(t-1)\left(\sum_{ij}(\hat{s}^1_{ij})^2 + \sum_{kl}(\hat{s}^2_{kl})^2\right) + t\,(t-1)\left(\sum_{ij}(\hat{s'}^1_{ij})^2 + \sum_{kl}(\hat{s'}^2_{kl})^2\right) + 2\,t\,(1-t)\left(\sum_{ij}\hat{s}^1_{ij}\,\hat{s'}^1_{ij} + \sum_{kl}\hat{s}^2_{kl}\,\hat{s'}^2_{kl}\right) &\leqslant 0 \\
\Leftrightarrow t\,(t-1)\left(\sum_{ij}(\hat{s}^1_{ij})^2 + \sum_{kl}(\hat{s}^2_{kl})^2 + \sum_{ij}(\hat{s'}^1_{ij})^2 + \sum_{kl}(\hat{s'}^2_{kl})^2 - 2\sum_{ij}\hat{s}^1_{ij}\,\hat{s'}^1_{ij} - 2\sum_{kl}\hat{s}^2_{kl}\,\hat{s'}^2_{kl}\right) &\leqslant 0 \\
\Leftrightarrow \sum_{ij}(\hat{s}^1_{ij})^2 + \sum_{kl}(\hat{s}^2_{kl})^2 + \sum_{ij}(\hat{s'}^1_{ij})^2 + \sum_{kl}(\hat{s'}^2_{kl})^2 - 2\sum_{ij}\hat{s}^1_{ij}\,\hat{s'}^1_{ij} - 2\sum_{kl}\hat{s}^2_{kl}\,\hat{s'}^2_{kl} &\geqslant 0 \\
\Leftrightarrow \sum_{ij}(\hat{s}^1_{ij})^2 - 2\,\hat{s}^1_{ij}\,\hat{s'}^1_{ij} + (\hat{s'}^1_{ij})^2 + \sum_{kl}(\hat{s}^2_{kl})^2 - 2\,\hat{s}^2_{kl}\,\hat{s'}^2_{kl} + (\hat{s'}^2_{kl})^2 &\geqslant 0 \\
\Leftrightarrow \sum_{ij}(\hat{s}^1_{ij} - \hat{s'}^1_{ij})^2 + \sum_{kl}(\hat{s}^2_{kl} - \hat{s'}^2_{kl})^2 &\geqslant 0
\end{aligned}$$

Finalement, $\forall \mathbf{s},\mathbf{s}' \in \mathbb{R}^\eta$ et $\forall t \in [0,1]$, on a $J(t\,\mathbf{s} + (1-t)\,\mathbf{s}') \leqslant t\,J(\mathbf{s}) + (1-t)\,J(\mathbf{s}')$, donc J est convexe.

En revanche, J n'est pas strictement convexe car si $\exists \mathbf{s},\mathbf{s}' \in \mathbb{R}^\eta$, tels que $\mathbf{s} \neq \mathbf{s}'$, on peut avoir $\forall i,j \in 1..\eta_1$, $(s^1_{ij} - s'^1_{ij}) = \sum_{kl} r_{ik}\,r_{jl}\,(s^2_{kl} - s'^2_{kl})$.

□

Annexe B

Sensibilité aux paramètres de χ-Sim

Cette annexe fournit des graphes 3D complémentaires pour l'étude de la sensibilité de l'algorithme χ-Sim présenté dans le Chap. II. En effet, la sensibilité de l'algorithme par rapport à ces différents paramètres a été étudiée dans la Sec. IV-2.3, grâce à des courbes 2D pour des moyennes sur un grand nombre de paramètres, et ces nouveaux graphes 3D permettent de confirmer et de compléter l'analyse qui a été faite.

L'ensemble des figures suivantes présentent la variation de précision micro-moyennée de l'algorithme χ-Sim, en fonction de ses trois paramètres T, k et p. Pour chacune des figures, un de ces paramètres est fixé, et les deux autres varient afin d'évaluer la robustesse de l'algorithme. Les jeux de données utilisés sont M5, M10, NG2, NG3 et la vue V_2 (traduction française) de ReutersEN (voir la Sec. IV-1 pour plus de détails).

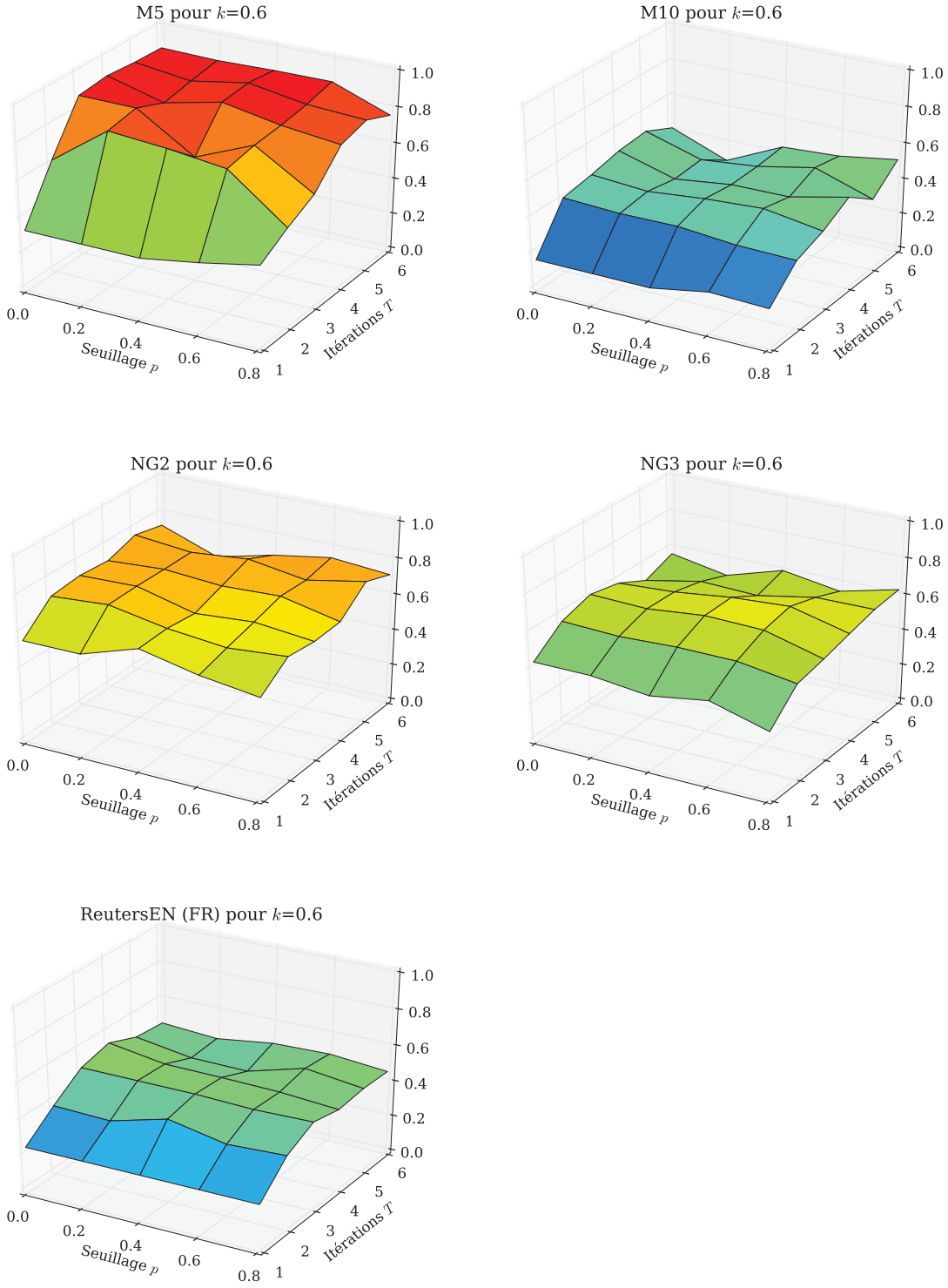


Figure 32 – Variation des performances de χ-Sim en fonction du nombre d'itérations T et du paramètre de seuillage p.

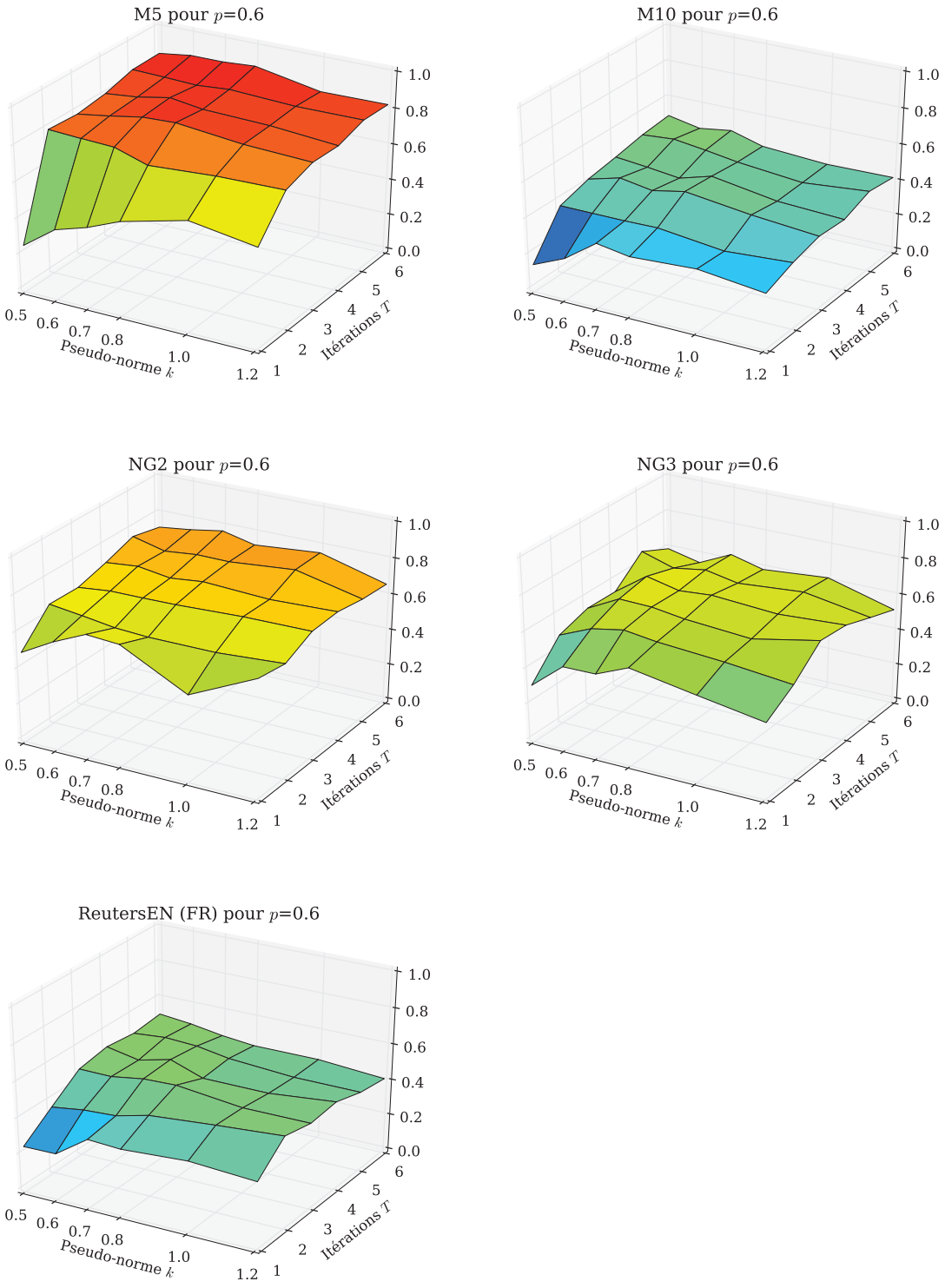


Figure 33 – Variation des performances de χ-Sim en fonction du nombre d'itérations T et de la pseudo-norme k.

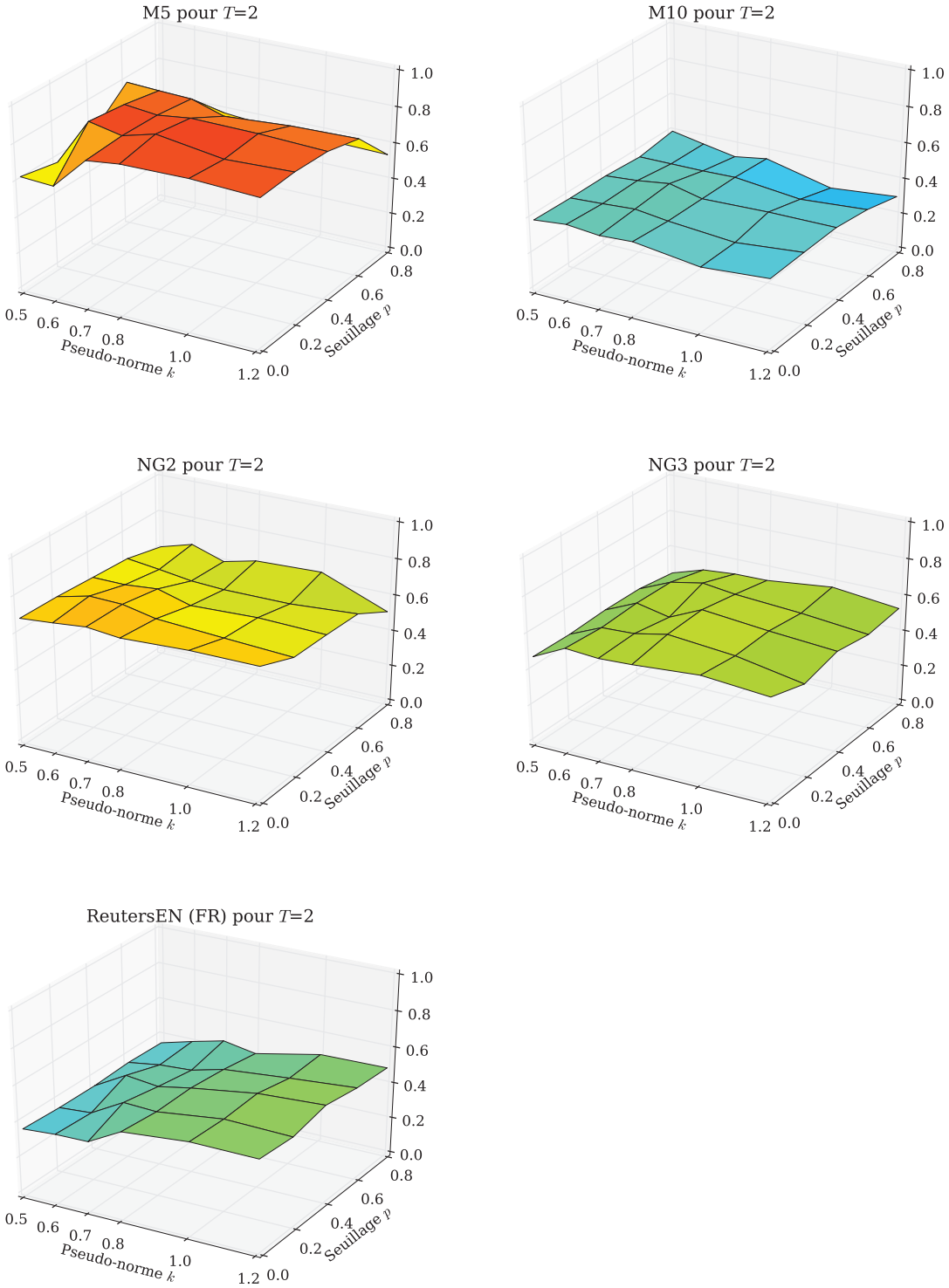


Figure 34 – Variation des performances de χ-Sim en fonction de la pseudo-norme k et du paramètre de seuillage p, pour $T = 2$.

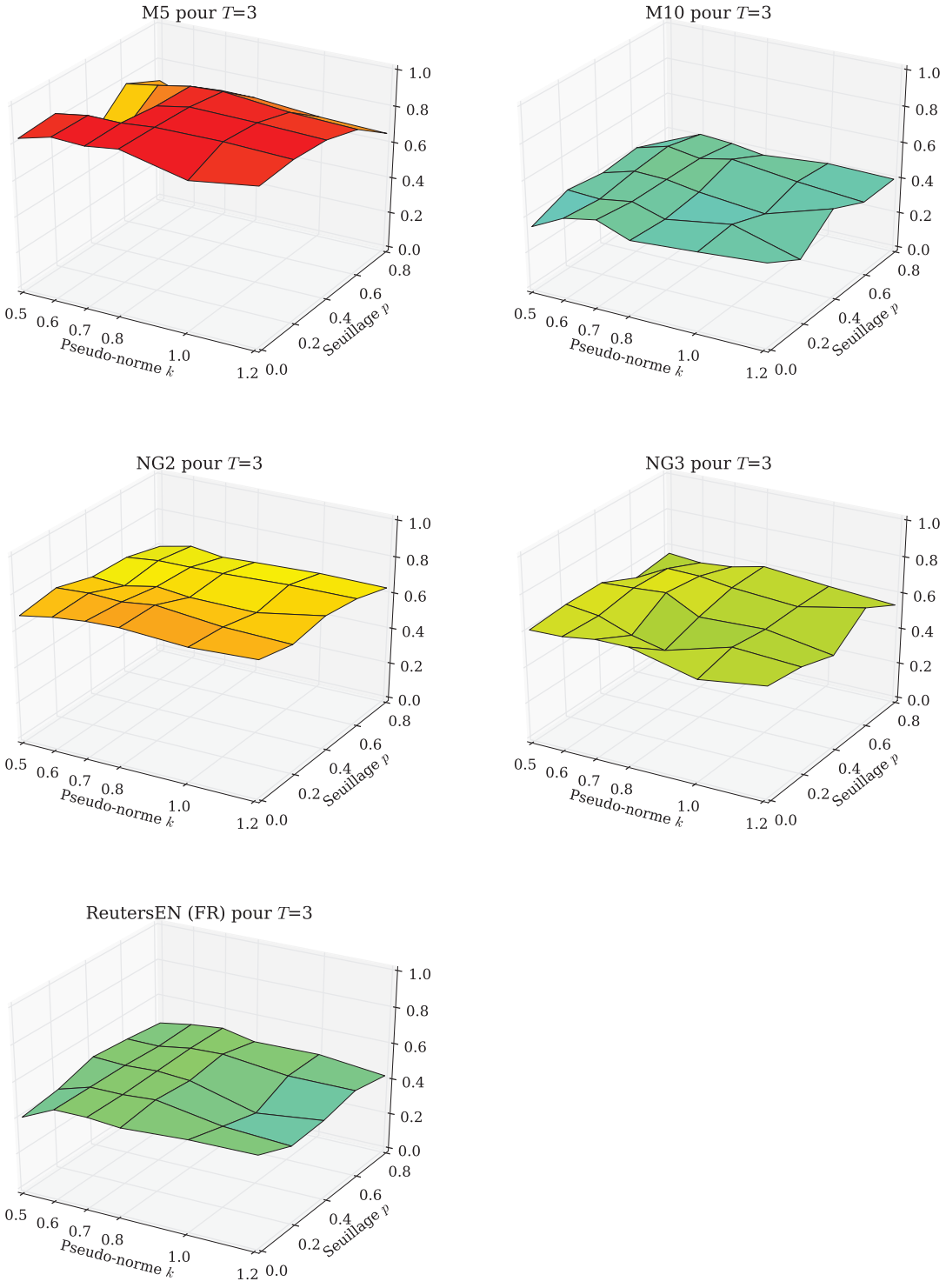


Figure 35 – Variation des performances de χ-Sim en fonction de la pseudo-norme k et du paramètre de seuillage p, pour $T = 3$.

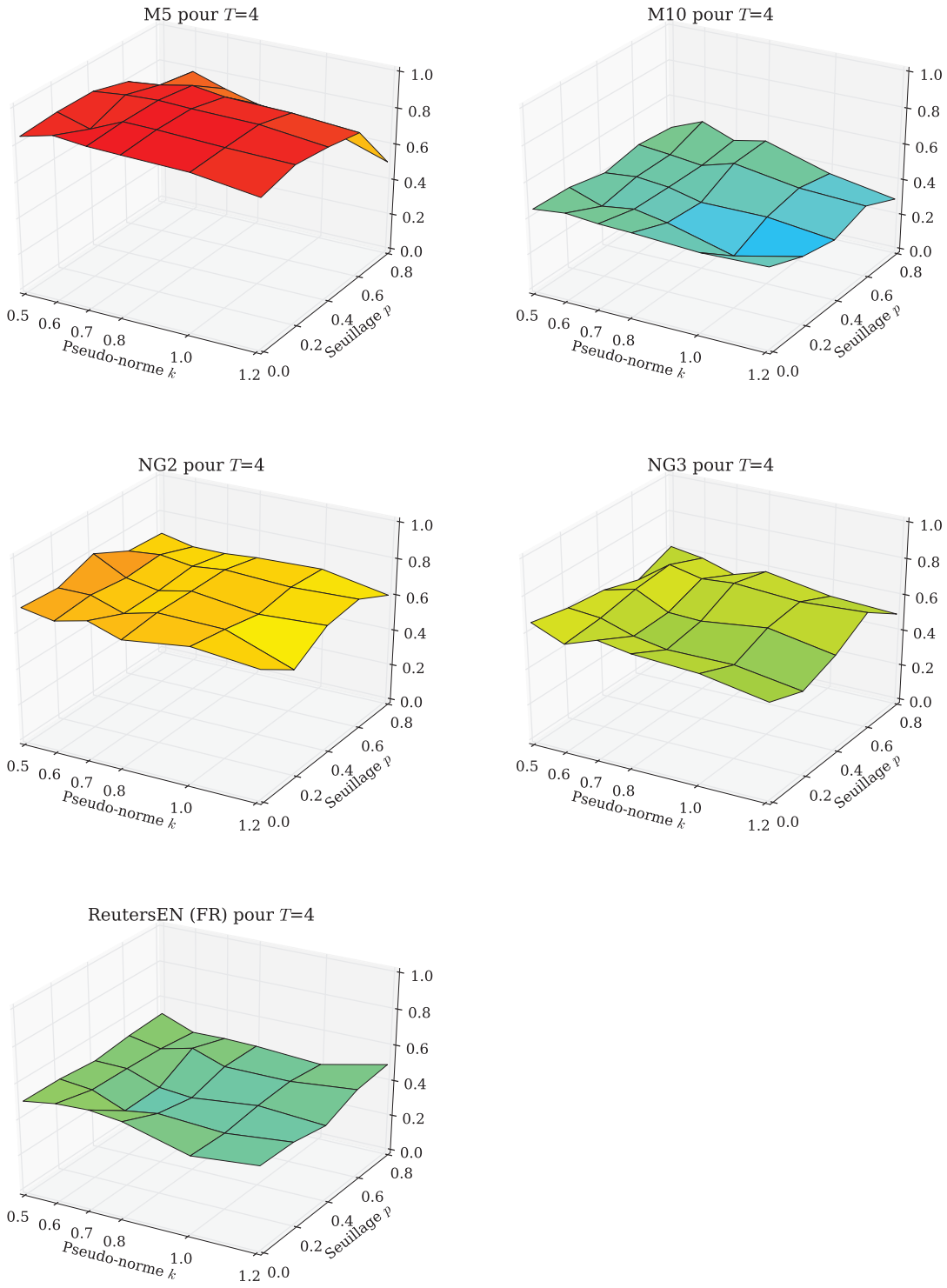


Figure 36 – Variation des performances de χ-Sim en fonction de la pseudo-norme k et du paramètre de seuillage p, pour $T = 4$.

Résumé

L'apprentissage automatique consiste à concevoir des programmes informatiques capables d'apprendre à partir de leurs environnements, ou bien à partir de données. Il existe différents types d'apprentissage, selon ce que l'on cherche à faire apprendre au programme, et également selon le cadre dans lequel il doit apprendre, ce qui constitue différentes tâches. Les mesures de similarité jouent un rôle prépondérant dans la plupart de ces tâches, c'est pourquoi les travaux de cette thèse se concentrent sur leur étude.

Plus particulièrement, nous nous intéressons à la classification de données, qui est une tâche d'apprentissage dit non supervisé. Dans ce type de tâche, le programme doit organiser un ensemble d'objets en plusieurs classes distinctes, de façon à regrouper les objets similaires ensemble. Dans de nombreuses applications, ces objets (des documents par exemple) sont décrits à l'aide de leurs liens à d'autres types d'objets (des mots par exemple), qui peuvent eux-même être classifiés. On parle alors de co-classification, et nous étudions et proposons dans cette thèse des améliorations de l'algorithme de calcul de co-similarités χ-Sim. Nous montrons que ces améliorations permettent d'obtenir de meilleurs résultats que les méthodes de l'état de l'art.

De plus, il est fréquent que ces objets soient liés à plus d'un autre type d'objets, les données qui décrivent ces multiples relations entre différents types d'objets sont dites multivues. Les méthodes classiques ne sont généralement pas capables de prendre en compte toutes les informations contenues dans ces données. C'est pourquoi nous présentons dans cette thèse l'algorithme de calcul multivue de similarités MVSim, qui peut être vu comme une extension aux données multivues de l'algorithme χ-Sim. Nous montrons que cette méthode obtient de meilleures performances que les méthodes multivues de l'état de l'art, ainsi que les méthodes monovues, validant ainsi l'apport de l'aspect multivue.

Finalement, nous proposons également d'utiliser l'algorithme MVSim pour classifier des données classiques monovues de grandes tailles, en les découpant en différents ensembles. Nous montrons que cette approche permet de gagner en temps de calcul ainsi qu'en taille mémoire nécessaire, tout en dégradant relativement peu la classification par rapport à une approche directe sans découpage.

Mots-clés : apprentissage automatique, classification, données multivues, similarité

Abstract

Machine learning consists in conceiving computer programs capable of learning from their environment, or from data. Different kind of learning exist, depending on what the program is learning, or in which context it learns, which naturally forms different tasks. Similarity measures play a predominant role in most of these tasks, which is the reason why this thesis focus on their study.

More specifically, we are focusing on data clustering, a so called non supervised learning task, in which the goal of the program is to organize a set of objects into several clusters, in such a way that similar objects are grouped together. In many applications, these objects (documents for instance) are described by their links to other types of objects (words for instance), that can be clustered as well. This case is referred to as co-clustering, and in this thesis we study and improve the co-similarity algorithm χ-Sim. We demonstrate that these improvements enable the algorithm to outperform the state of the art methods.

Additionally, it is frequent that these objects are linked to more than one other type of objects, the data that describe these multiple relations between these various types of objects are called multiview. Classical methods are generally not able to consider and use all the information contained in these data. For this reason, we present in this thesis a new multiview similarity algorithm called MVSim, that can be considered as a multiview extension of the χ-Sim algorithm. We demonstrate that this method outperforms state of the art multiview methods, as well as classical approaches, thus validating the interest of the multiview aspect.

Finally, we also describe how to use the MVSim algorithm to cluster large-scale single-view data, by first splitting it in multiple subsets. We demonstrate that this approach allows to significantly reduce the running time and the memory footprint of the method, while slightly lowering the quality of the obtained clustering compared to a straightforward approach with no splitting.

Keywords: machine learning, clustering, multiview data, similarity