Classification à partir d'une collection de matrices

Clément Grimal*, Gilles Bisson**

*Laboratoire d'Informatique de Grenoble, UMR 5217
Domaine Universitaire de Saint-Martin-d'Hères
38400 Saint Martin d'Hères, France
Clement.Grimal@imag.fr
**Laboratoire TIMC, CNRS / UJF 5525
Université de Grenoble - Domaine de la Merci
38710 La Tronche, France
Gilles.Bisson@imag.fr

Résumé. La classification simultanée de deux types d'objets – par exemple les mots et les documents dans les applications de recherche d'information – encore appelée *co-classification*, a été largement étudiée ces dernières années, et permet de découvrir la structure de ces objets, qu'elle soit explicite (classes) ou latente. Cependant, ce type d'approches ne permet de traiter qu'une seule relation entre deux types d'objets. Or, il existe de nombreux cas où des données multi-relationnelles sont disponibles, chaque paire d'objets pouvant être liée par une relation décrite par des données. Dans le cas des réseaux sociaux, on possède des données de co-occurrence décrivant les relations entre les acteurs et les différents objets. Dans cet article, nous décrivons un travail en cours ayant pour objectif d'exploiter ces données multi-relationnelles afin d'améliorer la qualité des classes obtenues. Ces méthodes sont une extension de l'algorithme de mesure de co-similarité χ-Sim développé par Bisson et Hussain (2008).

1 Introduction

L'objectif de la classification est d'organiser un ensemble d'*individus* suivant des critères de similarité, afin de retrouver, ou de découvrir la structure selon laquelle ils s'organisent. La classification a pour objectif de regrouper les objets dans des classes homogènes et contrastées : ainsi, la similarité d'un couple d'objets appartenant à la même classe doit être maximisée, tandis que celle d'une paire d'objets appartenant à deux classes différentes doit être minimisée. Les données utilisées, issues d'une collection d'observations, sont classiquement « non-structurées », mais représentables sous la forme d'une matrice dans laquelle les lignes correspondent aux *individus* et les colonnes aux *variables* les décrivant. Or, dans de nombreux

Ce travail a bénéficié d'une aide de l'Agence Nationale de la Recherche portant la référence ANR-08-CORD-009, dans le cadre du projet FRAGRANCES.

problèmes, les *variables* sont suffisamment homogènes pour que l'on puisse envisager de les catégoriser au même titre que les *individus*.

Récemment, le domaine de recherche visant à co-classifier deux types d'objets distincts a été intensivement exploré à la fois dans le domaine de la recherche documentaire, où le but est de co-classifier des documents selon la catégorie de sujets qu'ils traitent, et des mots qui les composent ; mais aussi dans le domaine de la bioinformatique afin, notamment, d'analyser les données d'expression génique.

Cependant, dans de nombreuses applications, les données de co-occurrence impliquant plus de deux types d'objets sont courantes, et les méthodes classiques de co-classification ne permettent pas de tirer partie de la richesse de cet aspect multi-dimensionnel. Nous nous proposons donc d'exploiter ces informations supplémentaires pour améliorer la classification des objets qu'elles décrivent. C'est dans ce contexte que nous travaillerons avec une collection de matrices de co-occurrence partageant certaines dimensions. Par exemple, ainsi que nous le verrons dans la partie 4, si le problème est de classifier des *films* selon leur genre, il peut être intéressant de prendre en compte simultanément plusieurs sources d'informations comme les matrices *films - acteurs* et *films - mots-clés*, qui partagent donc la dimension *films*, comme détaillé dans la partie 4.1.

L'objectif de notre méthode est alors de calculer les mesures de similarités entre les objets du même type, en utilisant toutes les données disponibles, le travail se base sur l'algorithme de calcul de co-similarité χ-Sim (Bisson et Hussain, 2008). En effet, l'un des intérêts de cette approche, outre les bons résultats obtenus sur la classification de données matricielles est qu'elle permet à l'utilisateur d'utiliser les matrices de similarité obtenues avec l'algorithme de classification qui lui convient le mieux (K-means, classification ascendante, classification par densité...) pour élaborer les classes d'objets.

L'article est organisé ainsi : dans la partie 2, nous décrivons les différentes classes de problèmes de co-classification, puis dans la partie 3, nous présentons nos extensions de la mesure de co-similarité χ-Sim au cas des données multi-relationnelles ; finalement, dans la partie 4, nous comparons nos mesures avec des méthodes courantes de (co-)classification.

2 Les différentes classes de problèmes

2.1 Bi-classification

Comme décrit dans l'introduction, lorsque les *variables* d'un problème de classification sont suffisamment homogènes, on peut choisir de les catégoriser au même titre que les *individus*. Dans le domaine de la recherche d'information, la co-classification apporte le double bénéfice d'améliorer la qualité des classes lorsque l'on travaille avec des matrices creuses et de grandes dimensions Long et al. (2005) ; mais également de mettre en évidence des ressemblance entre *individus* ayant des *variables*, a priori, très différentes. Ces ressemblances reposent sur l'hypothèse, dans le cas de la classification de textes, que deux documents appartenant à une même thématique, et donc appartenant à la même catégorie, peuvent certes contenir des termes différents, mais que ces termes doivent être a priori fréquents dans la collection des documents relatifs à cette thématique.

Il existe de nombreuses approches permettant de traiter cette classe de problèmes. Dans le domaine de la recherche documentaire, on peut citer entre autres les travaux de Dhillon (2001);

Liu et al. (2002); Dhillon et al. (2003); Long et al. (2005); Nadif et Govaert (2005); Rege et al. (2008); Bisson et Hussain (2008), ainsi que, dans le domaine de la bioinformatique, les travaux de Cheng et Church (2000); Madeira et Oliveira (2004); Speer et al. (2004).

En particulier, l'algorithme développé par Bisson et Hussain (2008) introduisant une nouvelle mesure de co-similarité χ-Sim, permet, en utilisant un algorithme de classification simple, de traiter cette classe de problèmes. C'est donc sur cet algorithme que nous baserons nos propositions dans la partie 3.

2.2 Classification de données multi-relationnelles

Les données issues d'applications réelles ont souvent des structures plus riches que celles traitées par les algorithmes de bi-classification, impliquant différents types d'objets, liés par des relations de co-occurrence, deux à deux. Par exemple, dans le cas de la classification de films selon leur genre, les *films* sont à la fois décrits par une liste d'*acteurs*, et par une liste de *mots-clés*. On peut donc mettre ces données sous la forme de deux matrices de co-occurrence, ayant une dimension commune, les *films*. Dans ce cas, l'objectif est donc de classifier simultanément les *films*, les *acteurs* et les *mots-clés*, en tenant compte simultanément de l'ensemble des relations de co-occurrence qui lient ces données. Il faut noter qu'il n'est pas trivial d'évaluer la classification de tous les types d'objets concernés (ici, les *acteurs* et les *mots-clés*), et l'utilisateur pourra ne pas être intéressé par le résultat de la classification de tous les types d'objets. Cependant, à l'instar de la bi-classification, la qualité de la classification de l'un de ces types d'objets (ici, les *films)*, pourra être grandement améliorée par cette approche.

Remarque 1 : *la représentation sous la forme d'une collection de matrices n'est pas la seule représentation valable. En effet, il est également possible de représenter ces données sous la forme de graphe k-partis* [1]*, chaque couche du graphe représentant un type d'objets différent Long et al. (2006).*

Bien que cette classe de problèmes soit plus récente que la précédente, et ait donc été moins fréquemment abordée, nous pouvons citer, entre autres, les travaux de Bekkerman et al. (2005); Wang et al. (2006); Long et al. (2006); Banerjee et al. (2007); Tang et al. (2009).

C'est cette classe de problèmes que nous traiterons dans la partie 3, en basant nos approches sur l'algorithme de calcul de co-similarité χ-Sim.

3 Algorithmes

3.1 Notations utilisées

De façon classique, les matrices sont notées en caractères majuscules gras, les vecteurs en minuscules gras et les autres variables en minuscules italiques.

Matrices de données : soit $\mathbf{M}_p$ l'ensemble des matrices contenant les informations de la base de données. Chaque matrice comporte r_p lignes (films) et c_p colonnes (acteurs ou mots-clés) : chaque valeur $m_{p,ij}$ caractérise *la nature de la relation* entre le $j^{\text{ème}}$ acteur (ou mot-clé) et le $i^{\text{ème}}$ film ; $\mathbf{m}_{p,i} = \left[m_{p,i1} \cdots m_{p,ic_p}\right]$ est le vecteur ligne correspondant au film i et $\mathbf{m}_p^j = \left[m_{p,1j} \cdots m_{p,r_pj}\right]$ est le vecteur colonne correspondant à l'acteur (ou mot-clé) j.

1. Un graphe est dit k-partis s'il existe une partition de son ensemble de sommets en k sous-ensembles, telle que deux sommets d'un même sous-ensemble ne puissent pas être adjacents.

Matrices de similarité : $\mathbf{SR}_p$ (de taille $r_p \times r_p$) et $\mathbf{SC}_p$ (de taille $c_p \times c_p$) représentent les matrices carrées et symétriques contenant respectivement les valeurs de similarité entre films et entre acteurs (ou mots-clés) avec $sr_{p,ij} \in [0,1]$, $1 \leqslant i,j \leqslant r_p$ et $sc_{p,ij} \in [0,1]$, $1 \leqslant i,j \leqslant c_p$. Le vecteur $\mathbf{sr}_{p,i} = \left[sr_{p,i1} \cdots sr_{p,ir_p}\right]$ (respectivement $\mathbf{sc}_{p,j} = \left[sc_{p,j1} \cdots sc_{p,jc_p}\right]$ indique la similarité entre le film i et l'ensemble des autres films (respectivement entre l'acteur, ou mot-clé j et l'ensemble des autres acteurs ou mots-clés).

Fonction de similarité : la fonction générique $\mathrm{F_s}(\cdot,\cdot)$ prend en entrée deux valeurs $m_{p,ij}$ et $m_{p,kl}$ de $\mathbf{M}_p$ et retourne la similarité entre ces valeurs avec $\mathrm{F_s}(m_{p,ij}, m_{p,kl}) \in [0,1]$; en outre, on pose l'hypothèse que $\mathrm{F_s}(m_{p,ij}, m_{p,kl}) = 0$ lorsque $m_{p,ij} = 0$ ou $m_{p,kl} = 0$.

3.2 L'algorithme de co-similarité χ-Sim

L'algorithme χ-Sim ne travaillant que sur une seule matrice de données, l'ensemble $\mathbf{M}_p$ se réduit à un seul élément que nous noterons $\mathbf{M}$, nous pouvons dans ce paragraphe simplifier les notations précédentes pour supprimer les indices p. Pour rappeler simplement le principe de cet algorithme de calcul de co-similarité χ-Sim (Bisson et Hussain, 2008), nous considérons que la matrice $\mathbf{M}$ est une matrice de co-occurrence *films-acteurs*, et où nous cherchons à calculer la matrice de similarité entre *films* $\mathbf{SR}$.

Dans le calcul de similarité entre les *films* $\mathbf{m}_i$ et $\mathbf{m}_j$, le principe fondamental de l'algorithme χ-Sim est de ne pas seulement considérer les *acteurs* jouant à la fois dans $\mathbf{m}_i$ et dans $\mathbf{m}_j$, mais bien l'ensemble de toutes les paires d'*acteurs* qui apparaissent dans ces *films*. De cette manière, l'algorithme capture non seulement les similarités des *acteurs* partagés entre $\mathbf{m}_i$ et $\mathbf{m}_j$, mais également les similarités des acteurs qui ne sont pas simultanément présents dans les deux *films*. La similarité entre deux *films* $\mathbf{m}_i$ et $\mathbf{m}_j$ est alors calculée par une fonction que nous noterons ici $\mathrm{Sim}(\mathbf{m}_i, \mathbf{m}_j)$ tel que :

$$\begin{aligned}
&\mathrm{Sim}(\mathbf{m}_i, \mathbf{m}_j) = \\
&\quad \mathrm{F_s}(m_{i1}, m_{j1}) \times sc_{11} + \mathrm{F_s}(m_{i1}, m_{j2}) \times sc_{12} + \ldots + \mathrm{F_s}(m_{i1}, m_{jc}) \times sc_{1c} + \\
&\quad \mathrm{F_s}(m_{i2}, m_{j1}) \times sc_{21} + \mathrm{F_s}(m_{i2}, m_{j2}) \times sc_{22} + \ldots + \mathrm{F_s}(m_{i2}, m_{jc}) \times sc_{2c} + \\
&\qquad \ldots \\
&\quad \mathrm{F_s}(m_{ic}, m_{j1}) \times sc_{c1} + \mathrm{F_s}(m_{ic}, m_{j2}) \times sc_{c2} + \ldots + \mathrm{F_s}(m_{ic}, m_{jc}) \times sc_{cc}
\end{aligned} \tag{1a}$$

Symétriquement, la similarité entre deux *acteurs* $\mathbf{m^i}$ et $\mathbf{m^j}$ est donnée par :

$$\begin{aligned}
&\mathrm{Sim}(\mathbf{m}^i, \mathbf{m}^j) = \\
&\quad \mathrm{F_s}(m_{i1}, m_{j1}) \times sr_{11} + \mathrm{F_s}(m_{i1}, m_{j2}) \times sr_{12} + \ldots + \mathrm{F_s}(m_{i1}, m_{jr}) \times sr_{1r} + \\
&\quad \mathrm{F_s}(m_{i2}, m_{j1}) \times sr_{21} + \mathrm{F_s}(m_{i2}, m_{j2}) \times sr_{22} + \ldots + \mathrm{F_s}(m_{i2}, m_{jr}) \times sr_{2r} + \\
&\qquad \ldots \\
&\quad \mathrm{F_s}(m_{ir}, m_{j1}) \times sr_{r1} + \mathrm{F_s}(m_{ir}, m_{j2}) \times sr_{r2} + \ldots + \mathrm{F_s}(m_{ir}, m_{jr}) \times sr_{rr}
\end{aligned} \tag{1b}$$

Cependant, les valeurs des éléments sr_{ij} de $\mathbf{SR}$ et sc_{ij} de $\mathbf{SC}$ doivent appartenir à l'intervalle $[0,1]$, or les valeurs calculées à partir des équations (1a) et (1b) ne vérifient pas cette propriété, ce qui implique une étape de normalisation. Par définition dans la partie 3.1, la fonction de similarité $\mathrm{F_s}$ renvoie des valeurs comprises entre 0 et 1, ainsi, la valeur maximale de

$\mathrm{Sim}(\mathbf{m}_i, \mathbf{m}_j)$ correspond au produit du nombre d'éléments non nuls de $\mathbf{m}_i$ et de $\mathbf{m}_j$. En notant ce produit $|\mathbf{m}_i| \times |\mathbf{m}_j|$, et en suivant le même raisonnement pour les éléments de $\mathbf{SC}$, les valeurs sr_{ij} de $\mathbf{SR}$ et sc_{ij} de $\mathbf{SC}$ peuvent se calculer ainsi :

$$\forall i,j \in 1..r, sr_{ij} = \frac{\mathrm{Sim}(\mathbf{m}_i, \mathbf{m}_j)}{|\mathbf{m}_i| \times |\mathbf{m}_j|} \qquad \forall i,j \in 1..c, sc_{ij} = \frac{\mathrm{Sim}(\mathbf{m}^i, \mathbf{m}^j)}{|\mathbf{m}^i| \times |\mathbf{m}^j|} \tag{2}$$

Lorsque la fonction de similarité $\mathrm{F_s}(m_{ij}, m_{kl})$ est définie comme le produit des deux valeurs m_{ij} et m_{kl}, le système d'équations permettant de calculer $\mathbf{SR}$ et $\mathbf{SC}$ peut être reformulé comme le produit de trois matrices. Ainsi, $\mathbf{SR}$ se calcule de cette manière [2] :

$$\mathbf{SR} = \left(\mathbf{M} \times \mathbf{SC} \times \mathbf{M}^T\right) \circ \mathbf{NR} \quad \text{avec } nr_{ij} = \frac{1}{|\mathbf{m}_i| \times |\mathbf{m}_j|} \tag{3a}$$

Symétriquement, $\mathbf{SC}$ se calcule ainsi :

$$\mathbf{SC} = \left(\mathbf{M}^T \times \mathbf{SR} \times \mathbf{M}\right) \circ \mathbf{NC} \quad \text{avec } nc_{ij} = \frac{1}{|\mathbf{m}^i| \times |\mathbf{m}^j|} \tag{3b}$$

Comme les équations décrites dans (2) ne sont pas indépendantes, il faut utiliser une méthode itérative pour calculer $\mathbf{SR}$ et $\mathbf{SC}$. Les deux matrices de similarité sont initialisées avec la matrice identité $\mathbf{I}$, soit $\mathbf{SR}^{(0)} = \mathbf{I}_r$ et $\mathbf{SC}^{(0)} = \mathbf{I}_c$. En effet, on considère qu'au départ, sans aucune connaissance a priori sur les données, la similarité entre un objet et lui-même vaut 1, et la similarité entre deux objets distincts est nulle. Puis, on calcule tour-à-tour $\mathbf{SR}$ en utilisant la matrice de similarité $\mathbf{SC}$ de l'itération précédente, et $\mathbf{SC}$ en utilisant la matrice de similarité $\mathbf{SR}$ de l'itération précédente. À chaque étape, les diagonales de $\mathbf{SR}$ et $\mathbf{SC}$ sont forcées à 1 afin de contraindre la similarité entre un objet et lui-même au maximum.

Finalement, il est intéressant de s'interroger sur le nombre d'itérations nécessaires pour calculer les valeurs finales de similarité. Dans cet algorithme, la notion d'itération a une forte signification car itérer n fois correspond à prendre en compte le $n^{\text{ème}}$ degré de co-occurrence entre les *films* et les *acteurs*.

3.3 Extensions aux données multi-relationnelles

On cherche maintenant à utiliser l'algorithme précédent permettant de calculer les co-similarités de deux types d'objets, pour calculer des co-similarités de différents types d'objets liés par des relations liant un type d'objets à un autre. Pour plus de simplicité, nous considérons par la suite le cas où nous cherchons à classifier des films selon leur genre (action, comédie...), connaissant pour chaque film, la liste des acteurs y apparaissant et la liste des mots-clés affectés par des utilisateurs. Nous possédons donc deux matrices de co-occurrence, une décrivant la relation *films - acteurs* notée $\mathbf{M}_1$ et une décrivant la relation *films - mots-clés* notée $\mathbf{M}_2$. L'objectif est donc d'examiner comment les informations contenues dans les matrices $\mathbf{M}_1$ et $\mathbf{M}_2$ peuvent être utilisées de manière conjointe et de vérifier que l'on obtient ainsi une classification des films, qui permette de retrouver les genres.

Dans les différentes approches, les « instances » de l'algorithme χ-Sim travailleront chacune avec une matrice de données distincte. On peut ainsi considérer que chaque « instance » de

2. L'opérateur « $\circ$ » désigne le produit de Hadamard, défini tel que, pour $\mathbf{C} = \mathbf{A} \circ \mathbf{B}$, les éléments de $\mathbf{C}$ sont calculés ainsi : $c_{ij} = a_{ij} \times b_{ij}$.

χ-Sim, travaillant avec sa propre matrice de données, est une *vue* différente du problème, et que l'objectif commun de toutes ces approches et de faire communiquer ces *vues* afin de tirer partie de toutes les informations disponibles.

3.3.1 Structure en cascade

Le principe de cette méthode est de calculer une première matrice de similarité des films à partir d'une des deux matrices de co-occurrence, puis de l'utiliser pour initialiser un second algorithme utilisant l'autre matrice de co-occurrence. On effectue n_1 itérations avec la première « instance » de χ-Sim, puis on initialise $\mathbf{SR}_2^{(0)}$ avec $\mathbf{SR}_1^{(n_1)}$, et on effectue n_2 itérations avec la seconde « instance ». Finalement, on utilise $\mathbf{SR}_2^{(n_2)}$ pour classifier les films.

Le schéma présenté sur la figure 1 présente un exemple de structure en cascade utilisant 2 « instances » fe χ-Sim.

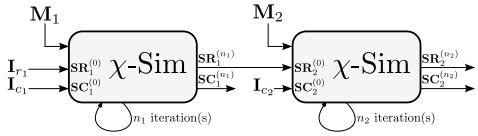

FIG. 1 – *Schéma de la méthode de structure en cascade. La matrice notée* $\mathbf{I}_{r_1}$ *représente la matrice identité de taille* r_1

Ainsi, lorsque la seconde «instance» de χ-Sim est utilisée, elle bénéficie des connaissances sur la similarité entre les objets partagés (ici les films), provenant de la première *vue*, qui doivent permettre d'améliorer les similarités calculées. On peut aisément généralisé cette structure à plus de deux « instances » de l'algorithme χ-Sim.

3.3.2 Structure en anneau

Le principe de cette méthode, utilisant une structure en anneau est de réaliser une seule itération avec chacune des matrices de données puis d'utiliser la matrice de similarité sur les lignes obtenue pour initialiser l'algorithme χ-Sim utilisant une autre matrice de données.

On alterne ainsi entre deux «vues» sur les données, ce qui peut permettre de faire circuler les informations de similarité de l'une à l'autre.

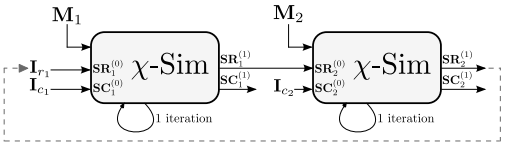

FIG. 2 – *Schéma de la méthode de structure en anneau. La matrice notée* $\mathbf{I}_{r_1}$ *représente la matrice identité de taille* r_1

Le schéma présenté sur la figure 2 représente le principe de cette méthode. À l'instar de la figure 1, on y voit deux «instances» de l'algotithme χ-Sim, la première utilisant la matrice de données $\mathbf{M}_1$, et la seconde utilisant $\mathbf{M}_2$. Chaque calcul de co-similarité à partir de l'une des matrices de données est donc utilisé par l'autre instance de χ-Sim par la suite.

3.3.3 Combinaison

Le principe de cette méthode est de calculer séparément deux matrices de similarité du même type d'objet, ici les films, puis de «combiner» ces deux matrices avant d'effectuer la classification des films. Différentes stratégies pour combiner ces matrices de similarité sont envisagées, parmi lesquelles, la moyenne, le minimum et le maximum des valeurs qu'elles contiennent.

Ainsi, en combinant les matrices de similarité obtenues, on unifie les différentes *vues* suivant un critère déterminé par la stratégie de combinaison (moyenne, minimum, maximum, etc.) choisie. Par exemple, en choisissant la stratégie du minimum, on considère que pour que deux films soient similaires, il faut qu'ils soient similaires dans les deux *vues* ; alors que pour la stratégie du maximum, il suffit qu'ils soient similaires dans au moins une des deux *vues*.

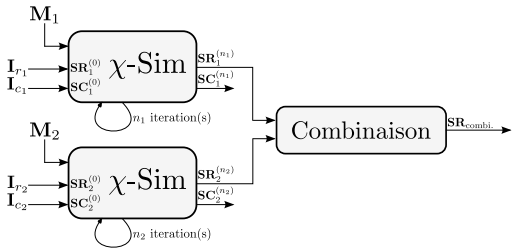

FIG. 3 – *Schéma de la méthode de combinaison. La matrice notée* $\mathbf{I}_{r_1}$ *représente la matrice identité de taille* r_1

Le schéma présenté sur la figure 3 représente le principe de cette méthode. On y voit deux «instances» de l'algorithme χ-Sim, la première utilisant ma matrice de données $\mathbf{M}_1$, et la seconde utilisant $\mathbf{M}_2$. De la stratégie de combinaison, représentée ici par le bloc **combi**., dépendent donc les résultats de cette méthode.

4 Expérimentations et résultats

L'évaluation de nos algorithmes a été réalisée à partir d'une série d'expérimentations sur un jeu de données où les *films* sont déjà catégorisés. Le critère d'évaluation consiste alors à mesurer l'adéquation entre les classes de films engendrées par un algorithme de classification utilisant la matrice de similarité $\mathbf{SR}$ produite par notre algorithme, et les catégories existantes. Nous évaluons donc ici l'apport de la co-similarité, et des données multi-relationnelles dans la classification des films par genre.

4.1 Base expérimentale

Les travaux dans le domaine de la classification de données multi-relationnelles étant encore peu nombreux, des jeux de données formés d'une collection de matrices sont peu fréquents. C'est pourquoi nous avons choisi de créer nos propres données, décrivant des films.

Nous avons utilisé *IMDb* [3] afin de recueillir le genre principal de chaque film [4] ainsi que la liste des acteurs y apparaissant et la liste des mots-clés affectés par des utilisateurs. Ainsi, nous avons collecté trois types d'objets différents : films, mots-clés et acteurs ; avec les deux relations suivantes : films - mots-clés et films - acteurs. Ceci nous donne donc deux matrices de co-occurrence, ayant la dimension commune *films*. De plus, il est important de noter que seule la qualité de la classification des films est évaluable, l'objectif étant de classer ensemble les films du même genre.

Les films choisis parmi la base de données de IMDb correspondent à un sous-ensemble de ceux utilisés dans le jeux de données *MovieLens* [5]. Nous avons cependant veillé à ce que les films retenus se répartissent de façon équilibrée dans les différents genres. Ainsi, les films de notre jeu de données ne représentent pas tous les genres d'IMDb mais les genres représentés possèdent tous un nombre similaires de films.

Par ailleurs, nous avons effectué un pré-traitement en supprimant les mots-clés utilisés pour moins de 3 films, les acteurs apparaissant dans moins de 2 films, et les films étant étiquetés par moins de 2 mots-clés ou ayant moins de 2 acteurs.

Finalement, notre jeux de données possèdent :

- 617 films de 17 genres différents (environ 36 films par genre)
- 1878 mots-clés
- 1398 acteurs

Chaque film est étiqueté par 33 mots-clés en moyenne, et une moyenne de 9 acteurs est à son casting. Un mot-clé est en moyenne utilisé pour étiqueter 11 films et un acteur apparait dans 4 films en moyenne.

4.2 Algorithmes utilisés

Nous avons comparé nos algorithmes aux mesures de similarité suivantes :

- *La similarité de cosinus.* Bien que cette mesure ne soit pas liée à la notion de co-classification, nous l'avons retenue car elle est intensivement utilisée en recherche d'information. Elle permet donc de définir une « ligne de base » pour les résultats et de mieux estimer l'apport d'une analyse conjointe des films et des acteurs/mots-clés, même si, comme dans cette expérimentation, on ne s'intéresse qu'à la seule classification des films.
- *La mesure de similarité induite par LSA (Latent Semantic Analysis).* Développée par Deerwester et al. (1990), elle est très souvent utilisée en recherche d'information entre autres. LSA permet d'établir des ressemblances entre objets ne partageant pas explicitement des occurrences avec d'autres objets du même type Kontostathis et Pottenger (2004). Il est donc intéressant de comparer une classification basée sur cette approche avec nos méthodes d'autant plus que les mécanismes sous-jacents sont différents : LSA est basée sur la décomposition en valeurs singulières de la matrice de donnée $\mathbf{M}$.
- *L'algorithme de co-classification ITCC (Information Theoretic Co-Clustering)* de Dhillon et al. (2003).

3. http://www.imdb.com/interfaces/

4. Dans IMDb, chaque film se voit attribuer plusieurs genres, mais celles-ci sont cependant hiérarchisées. Nous avons donc décidé de garder le premier comme genre.

5. http://www.grouplens.org/node/73

– *La co-similarité χ-Sim.* Cette mesure est quand à elle liée à la notion de co-classification et elle va nous permettre de démontrer l'apport des matrices de co-occurrence supplémentaires.

Il est évidemment nécessaire pour construire des classes de documents d'utiliser conjointement à ces mesure un algorithme de classification. Dans tous les cas, nous avons retenu l'algorithme de Classification Ascendante Hiérarchique (CAH) avec l'indice de Ward (1963), car, comme l'ont montré Bisson et Hussain (2008), cette méthode est celle qui permet d'obtenir les meilleurs résultats.

4.3 Implémentation et critère d'évaluation

Pour effectuer les tests, χ-Sim ainsi que nos algorithmes ont été implémentés en Java, en utilisant la bibliothèque JAMA [6] pour faciliter la manipulation des matrices. De même, les classifications ascendantes hiérarchiques ont été implémentées en Java.

De nombreuses mesures ont été proposées pour quantifier la ressemblance entre les classes initiales et les classes produites par un algorithme de classification. Elles sont basées sur l'analyse de la matrice de confusion $\mathbf{C}$ dans laquelle chaque élément c_{ij} indique le nombre de films de la classe réelle i qui ont été affectés à la classe apprise j (idéalement on devrait obtenir une matrice diagonale). Pour évaluer les résultats et permettre la comparaison avec les autres travaux nous avons utilisé l'indice classique suivant : la « précision micro-moyennée » ou micro-averaged Precision (Pr) introduite par Dhillon et al. (2003), pour laquelle une valeur de 1 correspond à une classification idéale.

4.4 Résultats expérimentaux

Pour tous les algorithmes disponibles, le test a porté sur l'intégralité de la base, il n'y a donc pas de valeur moyenne, ni d'écart-type disponible.

Nous avons utilisé deux matrices de données pour tester nos méthodes de calcul de mesure de co-similarité :

– $\mathbf{M_1}$: matrice de co-occurrence *films - mots-clés*
– $\mathbf{M_2}$: matrice de co-occurrence *films - acteurs*

De plus, dans ce cas particulier où les deux matrices ont la dimension *films* (les lignes) en commun, il est possible de juxtaposer ces deux matrices. On peut ensuite y appliquer les méthodes de bi-classification classiques. Bien que non générale, nous expérimentons cette stratégie et présentons les résultats dans la dernière ligne du tableau 1.

Pour LSA, nous avons fait varier le nombre k de valeurs singulières conservées dans l'intervalle $[10 \dots 200]$ avec un pas de 10 afin de trouver le nombre optimal de dimensions. La valeur rapportée pour chaque matrice de données est celle correspondant à la représentation des données et la valeur de k qui maximise la précision micro-moyennée.

Pour ITCC, nous avons fait varier pour chaque ensemble de tests le nombre de classes de mots dans la plage de valeur suggérée par Dhillon et al. (2003). Par ailleurs, comme ITCC repose sur une initialisation aléatoire des classes, chaque problème a été soumis 3 fois. Comme pour LSA les résultats rapportés correspondent aux paramètres qui maximisent la valeur moyenne de Pr.

6. http://math.nist.gov/javanumerics/jama/

Les tableaux 1, 2 et 3 présentent respectivement les résultats obtenus avec les méthodes classiques de mesures de co-similarité, les méthodes utilisant la structure en cascade et en anneau. Les résultats sur la méthode de combinaison sont omis pour des raisons de place.

Méthode	Cosinus	LSA	ITCC	χ-Sim	
				2 itération(s)	3 itération(s)
$\mathbf{M}_1$	0,225	0,277	0,280	0,256	**0,282**
$\mathbf{M}_2$	0,212	0,216	0,160	0,214	**0,217**
$\mathbf{M}_1\mathbf{M}_2$	0,284	**0,295**	0,266	0,261	0,280

TAB. 1 – *Résultats des expérimentations en terme de précision micro-moyennée (Pr) pour les tests issus du jeux de données décrit en 4.1 utilisant les méthodes classiques.*

	$\mathbf{M}_1 \rightarrow \mathbf{M}_2$	$\mathbf{M}_2 \rightarrow \mathbf{M}_1$
1 itération(s) $\rightarrow$ 1 itération(s)	0,207	0,254
1 itération(s) $\rightarrow$ 2 itération(s)	**0,227**	0,241
1 itération(s) $\rightarrow$ 3 itération(s)	0,222	0,285
2 itération(s) $\rightarrow$ 1 itération(s)	0,216	**0,292**
2 itération(s) $\rightarrow$ 2 itération(s)	0,227	0,246
2 itération(s) $\rightarrow$ 3 itération(s)	0,225	0,285

TAB. 2 – *Résultats des expérimentations en terme de précision micro-moyennée (Pr) pour les tests issus du jeux de données décrit en 4.1 utilisant la structure en cascade. Dans la première colonne,* 1 *itération(s)* $\rightarrow$ 2 *itération(s) signifie que l'on effectue 1 itération avec la première matrice, puis 2 itérations avec la seconde.*

Le tableau 2 regroupe l'ensemble des résultats permettant de mettre en évidence l'intérêt de la structure de cascade, il apparaît que les résultats obtenus en utilisant l'algorithme χ-Sim sur une seconde matrice de données sont en général meilleurs en utilisant la matrice de similarité obtenue à partir d'une première matrice de données, qu'avec la matrice identité. L'amélioration n'est cependant pas systématique et il serait intéressant de pouvoir déterminer à l'avance, l'apport d'une matrice de similarité donnée.

Notons également que le meilleur résultat que nous ayons obtenu à l'aide de cette structure en cascade donne une précision micro-moyennée de **0,292** avec deux itérations pour la première instance de χ-Sim et une itération pour la seconde instance de χ-Sim.

Le tableau 3 regroupe l'ensemble des résultats obtenus à l'aide de la structure en anneau.

	$\mathbf{M}_1 \leftrightarrow \mathbf{M}_2$	$\mathbf{M}_2 \leftrightarrow \mathbf{M}_1$
3 itération(s)	**0,292**	0,224
4 itération(s)	0,219	**0,266**

TAB. 3 – *Résultats des expérimentations en terme de précision micro-moyennée (Pr) pour les tests issus du jeux de données décrit en 4.1 utilisant la structure en anneau.* $\mathbf{M}_1 \leftrightarrow \mathbf{M}_2$ *signifie que l'on commence par faire une itération avec* $\mathbf{M}_1$*, puis une avec* $\mathbf{M}_2$*, jusqu'a ce que le nombre total d'itérations figurés dans la première colonne du tableau soit atteint.*

Il apparaît que la structure en anneau permet d'obtenir de très bons résultats, mais, à l'instar de la structure en cascade, l'amélioration de ces résultats n'est pas systématique. Notons le meilleur résultat de **0,292** – obtenu avec la structure en cascade et la structure en anneau – qui reste cependant inférieur à celui obtenu par LSA sur la juxtaposition des deux matrices avec **0,295**.

5 Conclusion

Dans cet article, nous avons proposé différentes méthodes permettant de tirer partie des informations fournies par des jeux de données multi-relationnels, impliquant chacune deux types d'objets différents. Nos méthodes s'appuient toutes sur l'algorithme de calcul de co-similarité χ-Sim développée par Bisson et Hussain (2008) très efficace dans des applications de recherche d'information (bi-classification de documents et des termes qui les composent), et dans le domaine de la bio-informatique (bi-classification de gènes et de leurs expressions).

Les différentes « instances » de l'algorithme χ-Sim que nos méthodes utilisent, correspondent à plusieurs vues sur les données dont nous disposons, l'objectif est alors d'agréger ces vues afin de tirer profit au mieux, des données qu'elles apportent. Bien entendu, si les différentes matrices de données apportent des informations contradictoires sur les types d'objets que l'on cherche à classifier, nos propositions ne permettront pas d'obtenir de meilleurs résultats qu'en utilisant une seule matrice de données.

Nous souhaitons continuer à expérimenter nos propositions sur des données représentées par plus de deux matrices. Nous aimerions également tester nos algorithmes sur des jeux de données concernant d'autres domaines, tels que la biologie ou la recherche documentaire.

Références

Banerjee, A., S. Basu, et S. Merugu (2007). Multi-way clustering on relation graphs. In *SDM*. SIAM.

Bekkerman, R., R. El-Yaniv, et A. McCallum (2005). Multi-way distributional clustering via pairwise interactions. In *ICML '05 : Proceedings of the 22nd international conference on Machine learning*, New York, NY, USA, pp. 41–48. ACM.

Bisson, G. et F. Hussain (2008). Chi-sim : A new similarity measure for the co-clustering task. In *Seventh International Conference on Machine Learning and Applications (ICMLA)*, pp. 211–217. IEEE Computer Society.

Cheng, Y. et G. M. Church (2000). Biclustering of expression data. In *Proceedings of the International Conference on Intelligent System for Molecular Biology*, Boston, pp. 93–103.

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

Dhillon, I. S. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD '01 : Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, New York, NY, USA, pp. 269–274. ACM.

Dhillon, I. S., S. Mallela, et D. S. Modha (2003). Information-theoretic co-clustering. In *Proceedings of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD-2003)*, pp. 89–98.

Kontostathis, A. et W. M. Pottenger (2004). A framework for understanding latent semantic indexing (lsi) performance.

Liu, X., Y. Gong, W. Xu, et S. Zhu (2002). Document clustering with cluster refinement and model. In *In Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 191–198. ACM Press.

Long, B., X. Wu, Z. M. Zhang, et P. S. Yu (2006). Unsupervised learning on k-partite graphs. In *KDD '06 : Proceedings of the 12th ACM SIGKDD*, New York, NY, USA, pp. 317–326. ACM.

Long, B., Z. M. Zhang, et P. S. Yu (2005). Co-clustering by block value decomposition. In *KDD '05 : Proceedings of the eleventh ACM SIGKDD*, New York, NY, USA, pp. 635–640. ACM.

Madeira, S. C. et A. L. Oliveira (2004). Biclustering algorithms for biological data analysis : A survey.

Nadif, M. et G. Govaert (2005). Block clustering of contingency table and mixture model. *Lecture notes in computer science 3646*, 249.

Rege, M., M. Dong, et F. Fotouhi (2008). Bipartite isoperimetric graph partitioning for data co-clustering. *Data Min. Knowl. Discov. 16*(3), 276–312.

Speer, N., C. Spieth, et A. Zell (2004). A memetic clustering algorithm for the functional partition of genes based on the gene ontology.

Tang, W., Z. Lu, et I. Dhillon (2009). Clustering with Multiple Graphs.

Wang, X., J.-T. Sun, Z. Chen, et C. Zhai (2006). Latent semantic analysis for multiple-type interrelated data objects. In *SIGIR '06 : Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 236–243. ACM.

Ward, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association 58*(301), 236–244.

Summary

Simultaneous clustering of two types of objects – words and documents for instance – also known as *co-clustering*, has been thoroughly studied in the past years, and allows discovering the structure, whether explicit or latent, of objects. However, many examples of real-world data involve objects of multiple types that are related to each other, and this kind of approaches is unable to take advantage of such data, available in various contexts, such as social networks. This article presents an ongoing work on extending the (co-)similarity measure χ-Sim developped by Bisson et Hussain (2008), in order co-cluster multi-type interrelated data objects.