Clustering de données relationnelles pour la structuration de flux télévisuels

Vincent Claveau*, Patrick Gros**

*IRISA – CNRS ** INRIA Rennes
Campus de Beaulieu, F-35042 Rennes
vincent.claveau@irisa.fr patrick.gros@inria.fr

Résumé. Les approches existantes pour structurer automatiquement un flux de télévision (i.e. reconstituer un guide de programme exact et complet), sont supervisées. Elles requièrent de grandes quantités de données annotées manuellement, et aussi de définir *a priori* les types d'émissions (publicités, bandes annonces, programmes, sponsors...). Pour éviter ces deux contraintes, nous proposons une classification non supervisée. La nature multi-relationnelle de nos données proscrit l'utilisation des techniques de *clustering* habituelles reposant sur des représentations sous forme attributs-valeurs. Nous proposons et validons expérimentalement une technique de *clustering* capable de manipuler ces données en détournant la programmation logique inductive (PLI) pour fonctionner dans ce cadre non supervisé.

1 Introduction

De nombreux services pour la télévision requièrent une segmentation et un étiquetage corrects du flux (corpus thématiques issus d'archives, TV à la demande...). Il faut ainsi disposer d'un guide de programme complet, documentant aussi les inter-programmes, et précis à l'image près. Un tel guide est malheureusement rarement disponible auprès des chaînes. Calculer ce guide de programme est le but de la structuration automatique des flux TV. Plusieurs approches ont été présentées dans la littérature pour ce faire. Qu'elles exploitent des métadonnées (Poli, 2008) ou des indices audio et vidéo (Naturel et Gros, 2008; Manson et Berrani, 2010; Ibrahim et Gros, 2011), toutes reposent sur une étape de classification supervisée nécessitant des connaissances *a priori* mais requièrent trop d'annotation manuelle pour être facilement utilisables en pratique. Par ailleurs, l'utilisateur doit définir les classes pertinentes pour un flux.

Dans cet article, nous proposons de réduire drastiquement l'intervention *a priori* de l'utilisateur en passant à une classification non supervisée. Le rôle résiduel de l'utilisateur serait alors d'étiqueter les classes qui émergent ainsi des données, plusieurs classes pouvant bien entendu partager la même étiquette. À l'image du célèbre K-MEANS, les techniques de clustering reposent sur une représentation simple des données, et une notion de distance entre ces représentations est également fournie par l'utilisateur (Jain, 2010). Ce sont ces deux points que nous cherchons à éviter. Depuis quelques années, certains travaux ont tenté de mettre à profit les capacités discrimantes des techniques d'apprentissage supervisé dans un cadre non-supervisé.

Leur principe commun est de déduire une similarité à partir de classifications identiques répétées sur des problèmes d'apprentissage factices (Shi et Horvath, 2005; Claveau et Ncibi, 2013). Elles permettent de ne pas avoir à expliciter la notion de distance entre données, mais reposent toujours sur une représentation classique des données sous forme attributs-valeurs qui n'est pas adaptée à la nature multi-relationnelle de nos données. Dans la veine de ces derniers travaux, nous proposons donc une technique de *clustering* capable de manipuler ces données. Plus précisément, nous détournons la programmation logique inductive (PLI) – technique d'apprentissage supervisé dont l'expressivité permet de représenter naturellement ce type de données – pour fonctionner dans ce cadre non supervisé.

2 Programmation logique inductive et données relationnelles

Que ce soit dans un cadre supervisé ou non supervisé, il est usuel de décrire les données à manipuler sous une forme propositionnelle, dite attribut-valeur, ou encore vectorielle. Les objets doivent alors tous avoir le même nombre d'attributs, et les attributs sont considérés indépendamment (les relations ne sont pas exploitées). Dans notre cas, les objets à manipuler sont des segments correspondant à des programmes ou des inter-programmes, dont beaucoup (e.g. les publicités) sont répétés plusieurs fois (appelées ci-après occurrences). Ce nombre d'occurrences est bien entendu différent d'un segment à l'autre, et chaque occurrence est elle-même décrite par un ensemble d'attributs. Cette variabilité rend la description sous une forme vectorielle impossible. De plus, certaines relations entre les attributs peuvent être particulièrement pertinentes, comme entre les chaînes de diffusion, les dates des différentes occurrences d'une émission ou des occurrences qui l'entourent. Cette nature multi-relationnelle (Dzerosky et Lavrac, 2001) des données est importante à prendre en compte pour assurer la pertinence du *clustering*.

La PLI est une technique d'apprentissage supervisé permettant d'inférer des règles générales H décrivant un concept à partir d'exemples E^+ et de contre-exemples de ce concept E^-, et d'un ensemble d'informations externes B appelé *Background Knowledge* (Muggleton et De Raedt, 1994, pour une présentation détaillée). La PLI est fondée sur la logique des prédicats : les données d'apprentissage sont décrites en Prolog et le classifieur est un ensemble de clauses de Horn. C'est cette expressivité qui permet de décrire les problèmes multi-relationnels. La figure 1 montre un bref extrait de description d'une émission dans B en Prolog standard. On y voit la façon simple de décrire les relations entre les différentes occurrences grâce aux prédicats binaires next_occ/2 et next_in_stream/2. Dans B sont aussi données les définitions de prédicats pouvant être utiles pour inférer les règles de H. Dans l'extrait précédent se trouvent les définitions du prédicat prev_occ/2, qui met en relation deux occurrences de la même émissions diffusées l'une après l'autre, et du prédicat interval/3 qui indique l'intervalle de temps entre deux occurrences de deux émissions.

Le but de la PLI est alors d'inférer un classifieur sous forme d'un ensemble H de clauses de Horn expliquant (c'est-à-dire impliquant ou couvrant) des exemples positifs et rejetant (le plus possible) les négatifs ; on le note $B, H \vdash E^+$ et $B, H \nvdash E^-$. Pour trouver les clauses composant H, on précise quels sont les prédicats définis dans B qui peuvent être utilisés et comment ils peuvent être combinés dans les clauses de H dans un langage d'hypothèse $\mathscr{L}_H$.

```
%%% description de la 1ère occurrence
has_occ(broadcast12,b12_occ1).      duration(b12_occ1,69).
date_time(b12_occ1,20,42,1,10,june,2005,friday).      channel(b12_occ1,2).
next_occ(b12_occ1,b12_occ2).      next_in_stream(b12_occ1,b28_occ5).
...
%%% connaissances générales
prev_occ(Occ1,Occ2) :- next_occ(Occ2,Occ1).
interval(Occ1,Occ2,Duration) :- date_time(Occ1,H1,Min1,S1,D1,M1,Y1,_),
   date2epoch(H1,Min1,S1,D1,M1,Y1,Epoch1), date_time(Occ2,H2,Min2,S2,D2,M2,Y2,_),
   date2epoch(H2,Min2,S2,D2,M2,Y2,Epoch2), Duration is abs(Epoch1-Epoch2).
...
```

FIG. 1: Extrait des descriptions des exemples et des connaissances sur le monde

Les règles composant H sont alors recherchées itérativement à travers un espace d'hypothèses (règles compatibles avec $\mathscr{L}_H$). Celles retenues sont celles maximisant un score, généralement défini en fonction du nombre d'exemples et de contre-exemples qu'elle couvre.

Voici un exemple de règle de H pouvant être inférée :

broadcast(A) :- has_occ(A,B), duration(B,3), next_occ(B,C), next_in_stream(B,D), next_in_stream(C,E), has_occ(F,D), has_occ(F,E).

Cette règle met bien en valeur l'intérêt d'une représentation multi-relationnelle : elle couvre toutes les émissions (variable A) ayant au moins deux occurrences qui se suivent (B, C), dont l'une a une durée de 3 secondes, elles-mêmes suivies par deux occurrences (D, E) d'une même émission (F). Cette règle couvrirait par exemple des émissions de sponsoring.

3 Du supervisé au non supervisé

3.1 Principes

L'idée principale de notre approche est de déduire une distance (ou une similarité) à partir de classifications répétées de deux émissions pour des tâches d'apprentissage aléatoire en PLI. Les émissions couvertes souvent par les mêmes clauses inférées seront supposées proches. L'algorithme 1 donne un aperçu global de la démarche. Comme pour le *bagging* (Breiman, 1996), la classification est répétée un grand nombre de fois en faisant varier les différents paramètres d'apprentissage : les exemples (étape 3 qui divise les données en exemples positifs E^+_{train} et en un ensemble E_{OoB} dit *out-of-bag* utilisé ensuite), les contre-exemples (étape 4), le langage d'hypothèse (étape 5). À chaque itération, un compte des paires d'émissions (x_i, x_j) couvertes par les mêmes clauses (on parle de *co-couvertures*) est tenu à jour dans la matrice $\mathcal{M}_{\text{co-couv}}$ en tenant compte des couvertures (une règle très discriminante "rapporte plus"). La dernière étape relève simplement de l'emploi d'une technique de clustering opérant sur cette matrice de co-couvertures, considérées comme des mesures de similarité. Dans nos expérimentations présentées dans la section suivante, nous utilisons le *Markov Clustering* (van Dongen, 2000). L'avantage par rapport au K-MEANS/K-MEDOIDS est de ne pas nécessiter de fixer *a priori* le nombre de clusters attendus, et d'éviter le problème de l'initialisation de ces clusters.

Algorithme 1 Clustering par PLI

input : E_{total} : émissions (non étiquetées)
for grand nombre d'itérations **do**
    E^{+}_{train}, $E_{\text{OoB}} \leftarrow$ Diviser(E_{total})
    Générer des exemples négatifs E^{-}_{train}
    Générer aléatoirement le langage d'hypothèse $\mathscr{L}_H$
    Inférence : $H \leftarrow$ PLI(E^{+}_{train},E^{-}_{train},$\mathscr{L}_H$)
    for all clause h_l parmi H **do**
        for all paire e_i, e_j de E_{OoB} telle que $B, h \vdash e_i, e_j$ **do**
            $\mathcal{M}_{\text{co-couv}}(e_i, e_j) += \frac{1}{|\{e_k | h_l, B \vdash e_k\}|}$
return Clustering($\mathcal{M}_{\text{coc-couv}}$)

La stratégie au cœur de cette approche est donc de varier les biais d'apprentissage à chaque itération. Le premier de ces biais est bien sûr l'ensemble d'exemples utilisés. Pour nos expériences, nous utilisons un dixième des exemples positifs tirés aléatoirement à chaque itération. C'est sur les 90 % restants que sont appliquées les règles inférées pour trouver les co-couvertures. La génération des exemples négatifs est un point important de l'algorithme. Il s'agit dans notre cas d'inventer des émissions, avec leurs différentes occurrences et leurs caractéristiques. Ces exemples doivent être suffisamment *réalistes* pour produire des tâches d'apprentissage dont la difficulté assure la pertinence des règles trouvées et donc des co-couvertures produites. Pour générer ces contre-exemples, nous copions des parties de descriptions d'émissions piochées aléatoirement dans B. Le format des règles autorisées, c'est-à-dire le langage d'hypothèse $\mathscr{L}_H$ est lui aussi différent à chaque tour. En pratique, tous les modes des prédicats possibles sont décrits à l'initialisation de l'algorithme, et un sous-ensemble (un tiers) est ensuite choisi aléatoirement à chaque itération. Ces apprentissages sur des tâches supervisées factices confèrent, par leur variété, des propriétés importantes à la similarité obtenue. Celle-ci mélange ainsi naturellement des descriptions complexes, opère par construction une sélection de caractéristiques, prend en compte les redondances des descripteurs, ignore ceux de mauvaise qualité, et elle est robuste aux données aberrantes.

4 Validation expérimentale

Évaluer une tâche de découverte de connaissances comme le *clustering* est toujours délicat, puisqu'elle suppose l'existence d'une vérité terrain dont on souhaite se passer. Les données que nous utilisons pour nos expériences celles de Naturel et Gros (2008) ; il s'agit d'un enregistrement de 22 jours consécutifs de la chaîne de TV France 2 de mai 2005. Ce flux est segmenté en émissions et les différentes répétitions (occurrences d'une même émission) ont été identifiées automatiquement puis consolidées manuellement (Naturel et Gros, 2008) et étiquetées selon six classes : programme, série, publicité, sponsoring, habillage de chaîne, bande-annonce. Ces classes d'émissions constituent notre vérité terrain, ou clustering de référence (cf. figure 2 pour leur répartition). Les mesures d'évaluation sont celles utilisées habituellement pour la comparaison de *clustering* (celui produit par notre approche vs. celui de référence) : l'Adjusted Purity, la Normalized mutual information et l'Adjusted Rand Index (ARI).

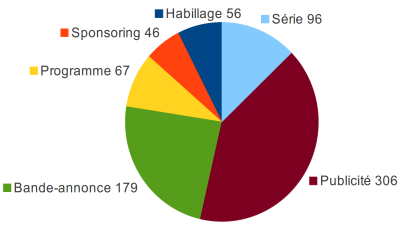

FIG. 2: Répartition des classes dans la vérité terrain.

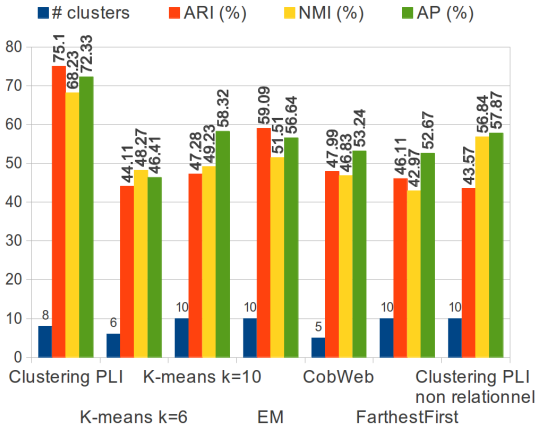

FIG. 3: Résultats des méthodes de clustering.

Nous présentons en figure 3 les résultats du *clustering* relationnel après 1 000 itérations ainsi que plusieurs *baselines* fondés sur une représentation classique des données, c'est-à-dire sous forme attribut-valeur. Les attributs utilisés pour décrire une émission sont les suivants : nombre d'occurrences, durée moyenne, intervalle minimal entre deux occurrences, intervalle maximal, intervalle moyen, nombre maximal d'occurrence sur une plage de 24h, durée entre la première et la dernière occurrence, présence ou non de toutes les occurrences dans la même journée et nombre moyen d'émissions uniques apparaissant avant ou après les occurrences. Les algorithmes *baseline* sont le K-MEANS, EM, CobWeb, tels qu'implémentés dans WEKA (Hall et al., 2009) ; pour chacun, nous ne rapportons que les résultats des meilleures configurations en terme d'ARI. Nous indiquons aussi les résultats de notre système avec cette description attribut-valeur (i.e. sans les prédicats relationnels de $\mathscr{L}_H$). Quelle que soit la mesure d'évaluation, notre technique de *clustering* offre de bien meilleurs résultats que les autres ; l'apport de la représentation apparaît clairement. Les clusters obtenus sont néanmoins différents en nombre et en contenu des classes attendues en vérité terrain. Une analyse des différences montre que la classe bande-annonce est difficile à capturer (elle est distribuée sur plusieurs clusters) ; d'autres cas problématiques sont causées par des émissions aux bornes de notre corpus ou pour lesquelles les trois semaines ne sont pas suffisantes pour identifier les schémas de récurrences.

L'examen des règles inférées à chaque itération permet aussi une validation indirecte de notre approche puisqu'elles exploitent bien l'aspect multi-relationnel de nos données. C'est le cas de la règle suivante qui couvre ainsi les émissions diffusées à intervalle régulier :

```
broadcast(A) :- has_occ(A,B), next_occ(B,C), next_occ(C,D), interval(B,C,E), interval(C,D,E).
```

5 Conclusions

La méthode de clustering que nous avons proposé nous permet bien de tirer au mieux profit de la nature particulières de nos données. Elle offre ainsi un moyen d'obtenir une notion de distance même dans des espaces de description riches et non métriques. Bien sûr, bien qu'il

n'y ait pas de définition explicite de la distance, d'autres biais induits par l'utilisateur sont présents, par exemple dans la description des données, la définition des modes possibles... Ces biais représentent la connaissance de l'utilisateur et permettent de définir son problème. L'apport de connaissances en terme de description des émissions sont autant d'informations permettant à l'utilisateur de canaliser la tâche de découverte sur son objet d'étude.

Références

Breiman, L. (1996). Bagging predictors. *Machine Learning 24*(2), 123–140.

Claveau, V. et A. Ncibi (2013). Découverte de connaissances dans les séquences par crf non-supervisés. In *Actes de la conférence TALN 2013*.

Dzerosky, S. et N. Lavrac (Eds.) (2001). *Relational Data Mining*. Berlin : Springer-Verlag.

Hall, M., E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, et I. H. Witten (2009). The WEKA data mining software : An update. *SIGKDD Explorations 11*(1), 10–18.

Ibrahim, Z. A. A. et P. Gros (2011). Tv stream structuring. *ISRN Signal Processing*.

Jain, A. K. (2010). Data clustering : 50 years beyond k-means. *Pattern Recognition Letters 31*(8), 651–666.

Manson, G. et S.-A. Berrani (2010). Automatic tv broadcast structuring. *International Journal of Digital Multimedia Broadcasting*.

Muggleton, S. et L. De Raedt (1994). Inductive Logic Programming : Theory and Methods. *Journal of Logic Programming 19-20*, 629–679.

Naturel, X. et P. Gros (2008). Detecting repeats for video structuring. *Multimedia Tools and Applications 38*(2), 233–252.

Poli, J.-P. (2008). An automatic television stream structuring system for television archives holders. *Multimedia Systems 14*(5), 255–275.

Shi, T. et S. Horvath (2005). Unsupervised learning with random forest predictors. *Journal of Computational and Graphical Statistics 15*(1), 118–138.

van Dongen, S. (2000). *Graph Clustering by Flow Simulation*. Thèse de doctorat, Université d'Utrecht.

Summary

The goal of automatic structuring of TV stream is to restore a complete and accurate program guide. Most approaches are supervised: they require large quantities of manually annotated data, but also to know a priori the different types of broadcasts (commercials, trailers, programs, sponsors...).. To avoid these constraints, we propose to replace this step by an unsupervised classification. The multi-relational nature of our data prohibits the use of usual clustering techniques based attribute-value representations. Thus, we propose a clustering technique capable of handling the data. We divert Inductive Logic Programming (ILP) to work unsupervised in this context and show the validity of the approach through different experiments.