Clustering multi-relationnal TV data by diverting supervised ILP

Vincent Claveau

Clustering multi-relationnal TV data by diverting supervised ILP

Vincent Claveau

IRISA - CNRS
Campus de Beaulieu, Rennes, France
vincent.claveau@irisa.fr

Abstract. Traditionally, *clustering* operates on data described by a fixed number of (usually numerical) features; this description schema is said propositional or attribute-value. Yet, when the data cannot be described in that way, usual data-mining or *clustering* algorithms are no longer suitable. In this paper, we consider the problem of discovering similar types of programs in TV streams. The TV data have two important characteristics: 1) they are multi-relational, that is to say with multiple relationships between features; 2) they require background knowledge external to their interpretation. To process the data, we use Inductive Logic Programming (ILP) [9]. In this paper, we show how to divert ILP to work unsupervised in this context: from artificial learning problems, we induce a notion of similarity between broadcasts, which is later used to perform the *clustering*. Experiments presented show the soundness of the approach, and thus open up many research avenues.

Keywords: clustering, diverting supervised technique, TV structuring

1 Introduction

Many TV services require the TV stream to be correctly segmented and tagged (thematic corpora from archives, TV on demand...). Thus, one needs a complete TV guide, also documenting inter-program (short spots between main programs, such as ads, trailers...), with a very high precision (at the frame level). Such guides usually do not exist, which makes their automatic building necessary. This task is at the heart of automatic structuring of TV streams. Several approaches have been proposed; some relies on meta-data [12] or audio/video clues [11, 8, 6]. They all rely on a supervised classification step (assign a class to each TV segment), thus requiring a priori knowledge (the user need to define the classes) and also too many manually annotated data to be actually usable. In this paper, we propose to reduce this important a priori involvement of the user by tackling the problem as a non-supervised one, that is as clustering. The remaining role of the user would then be to tag the clusters.

As with the well-known K-MEANS, clustering techniques rely on a simple representation of the data and on a distance notion operating of these representations which has to be provided by the user [7]. In our case, this leads to

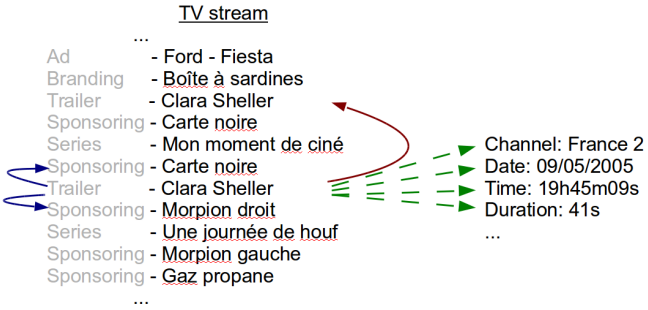

Fig. 1: Multiple relations of the trailer *Clara Sheller*.

two problems. First, our data need to be represented in a complex way, as they are multi-relational. Second, we do not know how to define a priori a relevant distance over these complex representations. In this paper, we propose to define a clustering technique suited to our complex data by diverting supervised Inductive Logic Programming (ILP) into a non-supervised technique. ILP makes it possible to easily represent our multi-relational data, and a distance between broadcasts is automatically from fake supervised classification problems, in the vein of [13, 2].

2 ILP and multi-relational data

For classification problems, objects are usually described in a propositional form, also said attribute-value or vector-based. In this representation, objects must have the same number of features, and the features are to be considered independently (relations between features are not exploited). In our case, each object is a segment of TV-streams corresponding to a program or an inter-program. But each object may have several occurrences, such as a particular ad which is repeated several times in the stream. The number of occurrences vary from one object to another, which makes the attribute-value description impossible. Moreover, certain relations between occurrences may be very relevant (eg. two occurrences are broadcast on different TV channels, two occurrences are broadcast in less than 1 day...). This multi-relational aspect of our data is thus important to consider for the clustering task. Figure 1 shows these different relations between occurrences and their feature as arrows with different colors (in gray: the class of broadcast, which is unknown in our problem).

ILP is usually used as supervised machine learning technique able to infer rules (eg. Horn clauses) H from examples (E^+) and counter-examples E^- of a concept, and with the help of background knowledge B [9]. Figure 2 shows how a program can be described in B (with standard Prolog). One can see how the relations between the occurrences are easily encoded with predicates next_occ/2 and next_in_stream/2.

```
%%% description of the 1st 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).
...
%%% other knowledge / predicate definition
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. 2: Excerpt of the example description and background knowledge

In B we also define the predicates that can be used to infer rules in H, such as prev_occ/2 which indicate two occurrences of the same program, one occurring after the other, or such as interval/3 wich indicates the time interval between two occurrences of two program. Here is an example of rule that can be inferred :
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).
This rule highlights the interest of the multi-relational representation: it covers every broadcast A having two occurrences B and C, lasting 3 seconds, such as these two occurrences are followed by two occurrences (D,E) from a same program (F). This rule typically covers sponsoring broadcast always appearing before a program.

3 From supervised to unsupervised

3.1 Principles

Our approach aims at deducing distances (or similarities) between two programs from repeated random classifications problems with ILP. For a given random classification problem, if the two programs are covered by H, it tends to show that they are related. If this is the case for every random classification problem, it means that they are very similar. Algorithm 1 gives an overview of the process. As for *bagging* [1], classification is repeated many times with different learning parameters: examples (step 3 wichich divides the data into positive E^+_{train} and E_{OoB}, a *out-of-bag* set used later), counter-examples (step 4), the hypothesis language (step 5). At each iteration, we record the pairs of programs (x_i, x_j) that are covered by the same inferred clauses (called co-covers hereafter) in a matrix $\mathcal{M}_{\text{co-cov}}$. One can give more weight to a clause covering very few examples, and less to a clause covering most of the examples (function weight). The last step is simply to use a standard clustering technique on a the co-cover matrix, considered as a similarity matrix. In the experiments presented below, we use

Algorithm 1 Clustering with ILP

input: E_{total}: programs
for i in [1 .. N] **do**
  E_i^+, $E_i^{\text{OoB}} \leftarrow$ Divide(E_{total})
  Generate negative examples E_i^-
  Generate randomly the hypothesis language $\mathscr{L}_i^H$ and the ILP parameters θ_i
  Inferring : $H_i \leftarrow$ ILP(E_i^+,E_i^-,$\mathscr{L}_i^H$,θ_i)
  for all clause h_l among H_i **do**
    for all pair e_a, e_b from E_i^{OoB} such that $B, h_l \vdash e_a, e_b$ **do**
      $\mathcal{M}_{\text{co-cov}}(e_a, e_b) += \text{weight}_i(e_a, e_b)$
    end for
  end for
end for
return *clustering*($\mathcal{M}_{\text{co-cov}}$)

Markov Clustering [3]. Its main advantage compared with K-MEANS/K-MEDOIDS is to avoid the need to decide *a priori* the number of expected clusters.

The strategy at the heart of this approach is to vary the learning biases at each iteration. The first bias is the set of examples used. In our experiments we use 1/10 of the programs to be used as positive examples. The inferred rules are then applied on the 9/10 remaining programs to find which one are co-covered. The generation of negative examples is an important step in our algorithm. In our case, it means inventing programs, with their occurrences and features. They have to be realistic enough in order to produce learning problems that will generate discriminative enough clauses, and thus relevant co-covers. In order to generate counter-examples, we randomly copy parts of the description of real programs (with a renaming of the constants in order to produce a coherent set of occurrences and features). The hypothesis language, setting the format of acceptable clauses, is also different at each iteration. In practice, every mode of every predicate is given at the initialization of the algorithm, and a subset is randomly chosen at each iteration.

4 Experiments

The data use for our experiments are those developed by [11]; it consists of a 22-day recording of the French France2 channel in May 2005. The stream is segmented in programs and the different occurrences of a same program have been identified automatically and manually consolidated [11]. To build the ground-truth needed to evaluate our clustering results, we used the manual annotation of the data proposed by [11] who tagged the programs according to 6 classes: movie/show, series, commercials, sponsoring, branding (short programs displaying the the name or logo of the channel), trailers (short programs announcing what will be broadcast later). This ground-truth tagging of the stream will be used as reference clusters (cf. Figure 3 for their repartition). The evaluation scores are those commonly used for clustering comparison (the one produced

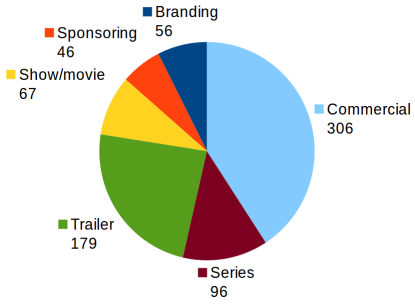

Fig. 3: Class repartition in the ground-truth.

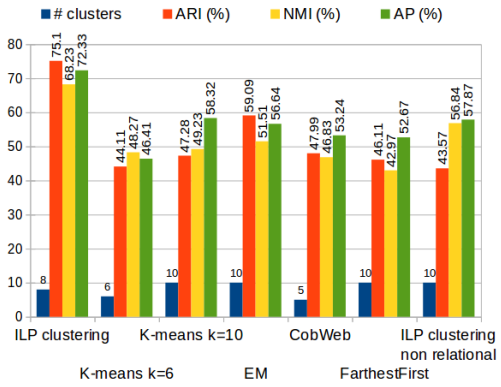

Fig. 4: Results of clustering methods.

automatically vs. the ground-truth one): Adjusted Purity, Normalized mutual information and Adjusted Rand Index (ARI) [15].

Figure 4 presents the results of the relational *clustering* after 1 000 iterations as well as several *baselines* relying on a usual propositional representation. In this latter case, the features used are: number of occurrences, average duration, mean, minimal and maximal interval between two occurrences, maximal number of occurrences during a 24h, duration between the first and last occurrences, presence or not of every occurrences in the same day, and average number of other programs occurring before or after the program occurrences. The baseline algorithms are: K-MEANS, EM, CobWeb, such that implemented in WEKA; for each of them, we only report the results of the configurations yielding the best ARI. The ILP algorithm used is ALEPH [14], the data are described as shown in Section 2. We also, report the results of our ILP-based approach exploiting the same representation (i.e. discarding the relational predicates of $\mathscr{L}_H$).

For any evaluation score, our ILP-based clustering approach perform better than the propositional approaches; it clearly shows the added-value of the ability to handle the multi-relational representation of the data. The generated clusters are nonetheless different in terms of numbers of clusters and in terms and of the content of these clusters. An analysis of the differences between the ILP clusters and ground-truth ones shows that the trailer class is difficult to capture (such programs appear in several ILP clusters). Other problems are caused by programs at the boundaries of our 22-day TV recording or for programs for which the 3 weeks are not enough to capture the recurrence patterns.

An analysis of the inferred rule for each iteration also allow an indirect validation of our approach since they exhibit the multi-relational property of our data. This is the case of the following rule which covers programs broadcast at fixed interval:

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

5 Conclusions

Our clustering approach, relying on ILP, allows us to make the most of the multi-relational aspect of our TV data. It makes it possible to get a notion of distance even in rich description spaces where metrics cannot be defined a priori. Of course, even if there is no explicit definition of the distance, other biases from the user are unavailable, such as the way the data are described, the definition of the modes in the hypothesis language...

Several perspectives are foreseen. For our TV application, the use of a larger dataset (recording several months with several channels) would allow us to limit the errors mentioned in the previous section. Adding multimodal features (logo detection, black frames, speech detection...) would also bring useful information about the content of the TV segment. These features should help the clustering process to distinguish between branding and sponsoring, or to better categorize trailers.

References

1. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
2. Claveau, V., Ncibi, A.: Découverte de connaissances dans les séquences par CRF non-supervisés. In: Actes de la conférence TALN 2013 (2013)
3. van Dongen, S.: Graph Clustering by Flow Simulation. Thèse de doctorat, Université d'Utrecht (2000), `\url{http://micans.org/mcl/}`
4. Dzerosky, S., Lavrac, N. (eds.): Relational Data Mining. Berlin: Springer-Verlag (2001)
5. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: An update. SIGKDD Explorations 11(1), 10–18 (2009)
6. Ibrahim, Z.A.A., Gros, P.: Tv stream structuring. ISRN Signal Processing (2011)
7. Jain, A.K.: Data clustering: 50 years beyond k-means. Pattern Recognition Letters 31(8), 651–666 (2010)
8. Manson, G., Berrani, S.A.: Automatic tv broadcast structuring. International Journal of Digital Multimedia Broadcasting (2010)
9. Muggleton, S., De Raedt, L.: Inductive Logic Programming: Theory and Methods. Journal of Logic Programming 19-20, 629–679 (1994)
10. Muggleton, S., De Raedt, L., Poole, D., Bratko, I., Flach, P.A., Inoue, K., Srinivasan, A.: ILP turns 20 - biography and future challenges. Machine Learning 86(1), 3–23 (2012), `http://dx.doi.org/10.1007/s10994-011-5259-2`
11. Naturel, X., Gros, P.: Detecting repeats for video structuring. Multimedia Tools and Applications 38(2), 233–252 (2008)
12. Poli, J.P.: An automatic television stream structuring system for television archives holders. Multimedia Systems 14(5), 255–275 (2008)
13. Shi, T., Horvath, S.: Unsupervised learning with random forest predictors. Journal of Computational and Graphical Statistics 15(1), 118–138 (2005)
14. Srinivasan, A.: The ALEPH manual. Machine Learning at the Computing Laboratory, Oxford University (2001)
15. Vinh, N.X., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison. Journal of Machine Learning Research (2010)