Clustering multi-relational TV data with ILP

Vincent Claveau

UMR IRISA

Problem

Tagging segments from TV streams

- desequencialize the stream into meaningful segments
- repeated segments are found [Naturel08]

Problem

Tagging segments from TV streams

- desequencialize the stream into meaningful segments
- repeated segments are found [Naturel08]

Goal

- discriminate between commercials, soaps, sponsoring...
  - based on 'basic' features (eg. length)...
  - ...but also on how they repeat, their neighbors
- cluster segments without a priori
  - no predefined classes, no annotated data

Problem

Representation limits with usual clustering

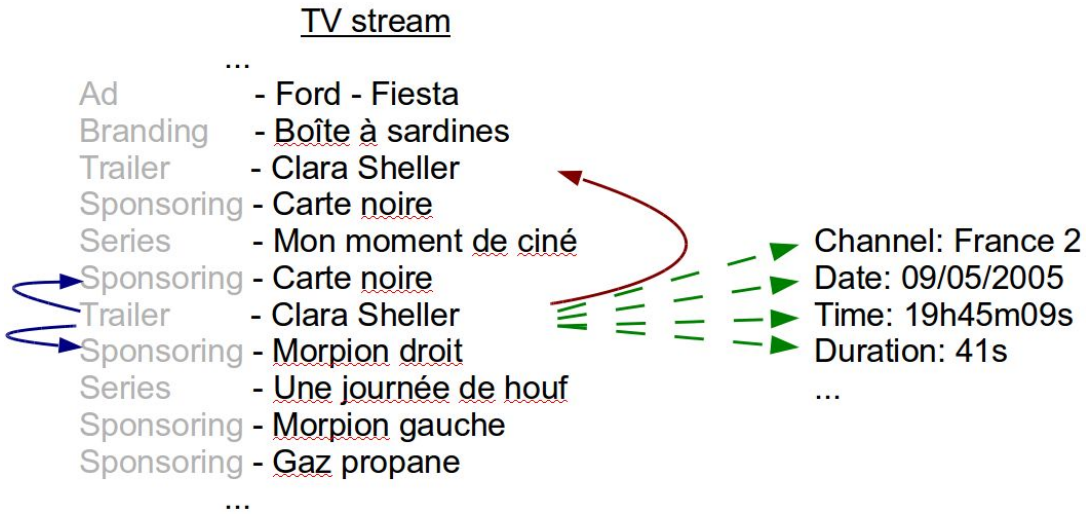
- relies on attribute-value description
  - objects are put into a multidimensional space (1 feature = 1 dimension)
- distances are computed, close objects are gathered...

	Feature 1	Feature 2	Feature 3
object 1	...	...	...
object 2	...	...	...

Problem

Representing TV data

- an object = multiple occurrences of a repeated segment
- an object has occurrences which have features
  - and also object has features
- variable number of occurrences, and thus of features



Problem

Representing TV data

- an object = multiple occurrences of a repeated segment
- an object has occurrences which have features
  - and also object has features
- variable number of occurrences, and thus of features

	date occ 1	duration occ 1	channel occ 1	date occ 2	duration occ 2	channel occ 2	... ...
Darty	ts 14503023	12s	2	ts 14504332	11s	2	...
Shrek	ts 14502455	94mn	2	?	?	?	...

Missing features

Problem

Representing TV data

- an object = multiple occurrences of a repeated segment
- an object has occurrences which have features
  - and also object has features
- variable number of occurrences, and thus of features

	...	delay occ 1/occ2	delay occ1/occ 3	same channel occ1/occ 2	...
Darty	...	24h05mn	48h02mn	yes	...
Shrek	...	*?*	*?*	*?*	...

Need to flatten every relation

Our approach

Idea 1: use Inductive Logic Programming

- ML technique based on FOL [Muggleton 95]
- infers rules H from pos and neg examples E^+, E^- and background knowledge B according to language L
  - $B, H \models E^+$ s.t. $H \in L$
- able to handle this kind of relational data [Dzerosky & Lavrac 01]
  - E^+ , E^- , B, H are expressed in Prolog

Description

Consider one (repeated) segment broadcast12

```
nb_occ( broadcast12, 3 ).

has_occ( broadcast12, b12occ1 ).

date_time( b12occ1, 20,42,1, 10,june,2005, friday ).

duration( b12occ1, 69 ).

channel( b12occ1, 2 ).

next_occ( b12occ1, b12occ2 ).

next_in_stream( b12occ1, b28 occ5 ).


has_occ( broadcast12, b12occ2 ).

date_time( b12 occ2, 23,48,19, 10,june,2005, friday ).
```

description of 1st occurrence

relation

relation

description of 2nd occurrence

Description

```
prev_occ(Occ1,Occ2) :- next_occ(Occ2,Occ1).

interval(Occ1,Occ2,Duration) :- datetime(Occ1,H1,Min1,S1,D1,M1,Y1,
_),
                    date2epoch(H1,Min1,S1,D1,M1,Y1,Epoch1),
                    datetime(Occ2,H2,Min2,S2,D2,M2,Y2,_),
                    date2epoch(H2,Min2,S2,D2,M2,Y2,Epoch2),
                    Duration is abs(Epoch1-Epoch2).
...
```

Our approach

Remember: no predefined classes, no annotated data

Idea 2: make ILP unsupervised

- can we contort ILP to work without supervision?
- generate many fake learning problems
  - make emergence distance between objects
  - inspired by Random Forests clustering [Shi & Horvath 05]

Our approach

1 divide the objects into 2 sets: E^{+}, E^{OoB}

2 generate fake negative examples E^{-}

3 select learning parameters (L, length, search options...) at random

4 run ILP with E^{+}, E^{-} and these parameters [Aleph, Srinavasan01]

5 remember objects in E^{OoB} that are covered by the same clauses from H

6 repeat steps 1-5 many times

Our approach

	broadcast 1	broadcast 2		broadcast n
broadcast 1	100	15	...	3
broadcast 2	15	100	...	23
...	...	...	100	...
broadcast n	3	23	...	100

- co-covering counts are interpreted as a similarity
- Markov clustering [van Dongen 00] on the obtained similarities

Our approach

Comments

- cannot prevent biases, so change them at each iteration
  - features used, the E − generated, random subset of E +
  - change any relevant parameters of the ILP algo
- similarity can be formally defined by

$$s^N(e_a, e_b) = \sum_{i=1}^{N} \mathbb{1}_{\{e_a, e_b \in E_i^{\text{OoB}} | \exists h_l \in H_i, B, h_l \vdash e_a, e_b\}}$$

- resulting similarity ~ mixture of 2 Poisson-binomial distributions [Royer et al. 2015]

Generating negative examples

Several strategies

- completely at random

  ☒ not realistic → too easy to discriminate pos from neg

- following the same distribution

  ☒ consider features independently, not easily feasible

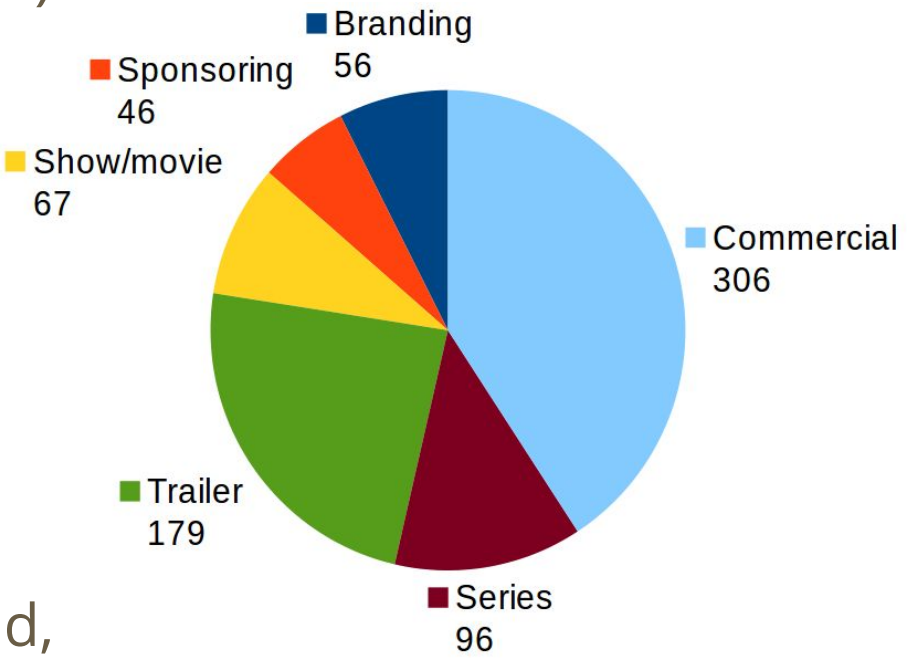
- copy characteristics of the positive examples

  ☑ copy random parts from E^+ description (renaming the object id) ensure to keep some realistic relational info

Experiments

Data

- 3-week long TV stream (France2)
- manually segmented and annotated [Naturel08]



Reference

- clusters proposed by experts
  - 2 types of 'plain' programs
  - 4 types of interprograms (ad, sponsoring, trailer, channel branding)

Experiments

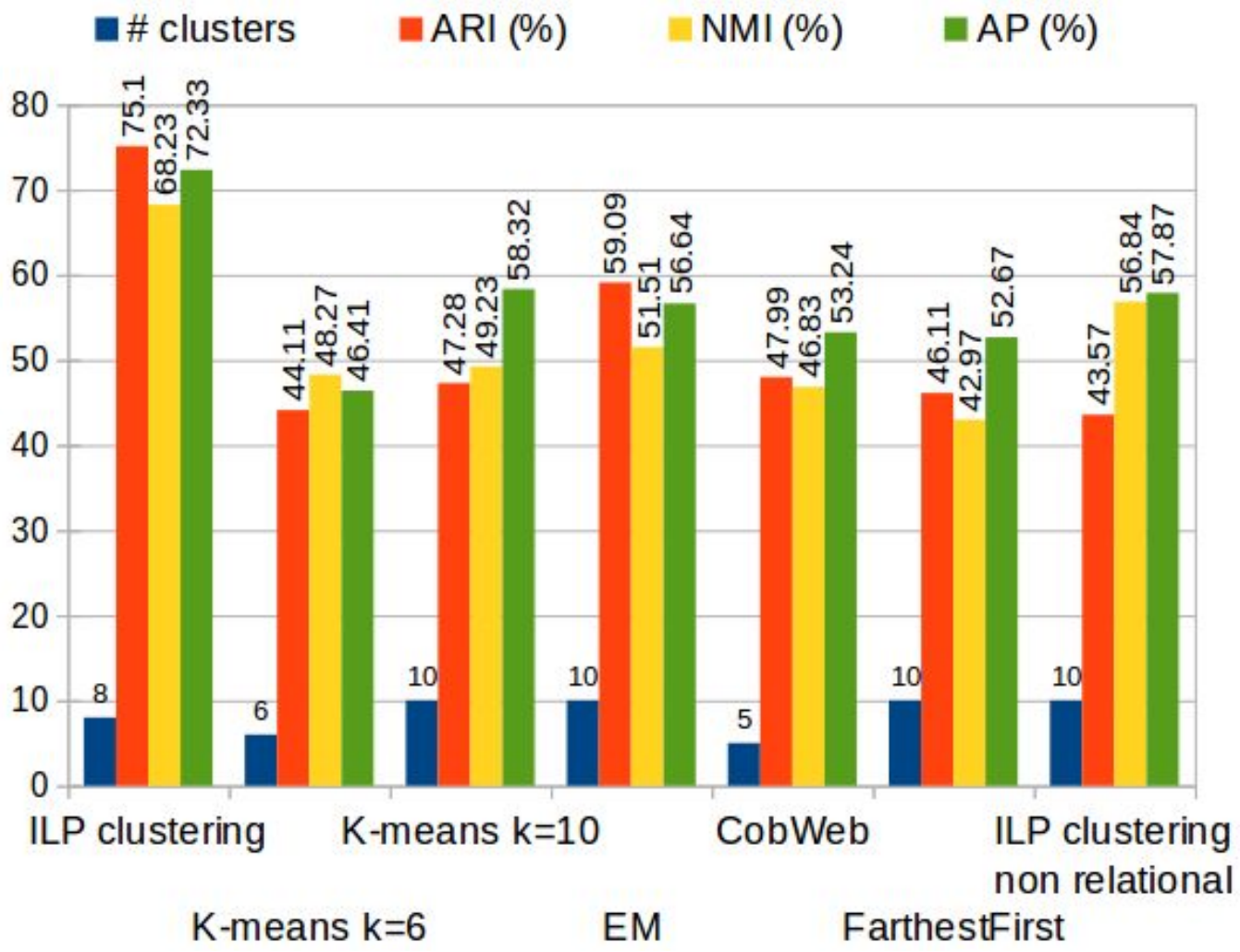
Evaluation

- comparison with the 'ground-truth' classes
- usual measures: Adjusted Purity, Normalized mutual information, Adjusted Rand Index

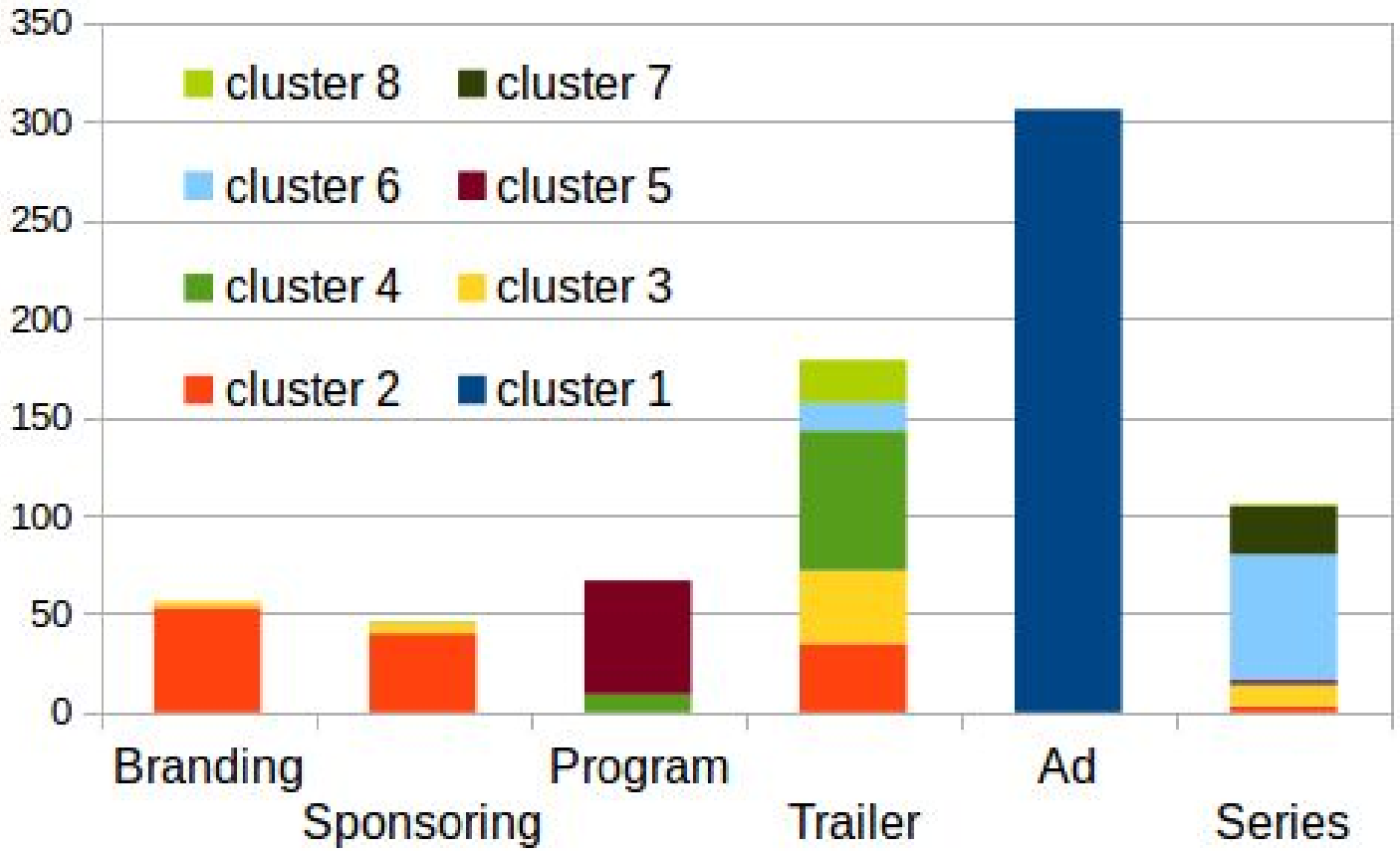
Baselines

- usual clustering techniques: k-means, EM, CobWeb, FarthestFirst [WEKA]
  - features: # occurrences, avg duration, max, min and avg interval between 2 occ, max # occ / 24h, avg nb of broadcasts after or before...
- ILP-based clustering without the relational information

Main results



Cluster analysis



Rule analysis

Rules based on propositional features

```
broadcast(A) :- nb occ(A,1).
```

Rules exploiting the multi-relational capacity of ILP

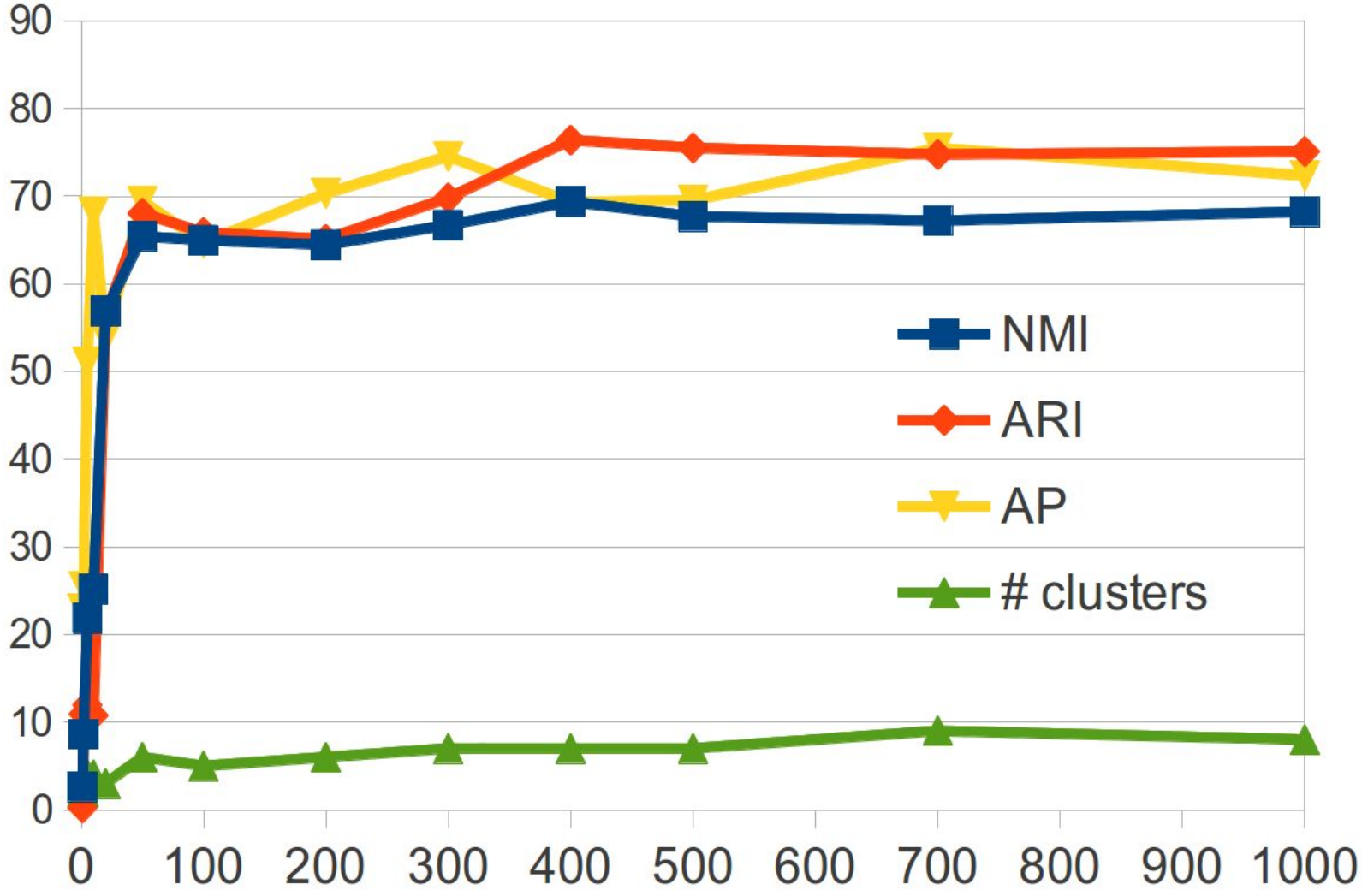
```
broadcast(A) :- has occ(A,B), next occ(B,C), next occ(C,D), interval(B,C,E), interval(C,D,E).
```

→ same interval between successive occurrences (~series)

```
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).
```

→ short duration + same neighbors for each occurrence

How many iterations?



Conclusions

Clustering relational data

- clustering is done by running fake supervised problems
  - as for bagging, random+multiple runs → robustness
- multi-relational nature of our data is handled by ILP

Problems

- ILP has a huge complexity, repeating the learning step leads to huge execution times
  - can be easily distributed: 1 node = 1 learning step
- ILP is not good at handling numeric data

Foreseen work

Real world settings

- automatic pre-processing: deal with noisy segments
- 2 × 6 months of TV on 2 channels: easier to discriminate ads
- semi-supervised clustering: embed expert knowledge on classes
  - 2 objects must or cannot belong to the same class
  - join/refine existing clusters

Other applications of our approach

- use CRF to cluster sequences [Claveau & Ncibi 13]
- use HMM to cluster speech signals