Transformation and aggregation preprocessing for top-k recommendation GAP rules induction

Marta Vomlelova, Michal Kopecky, Peter Vojtas

Faculty of Mathematics and Physics
Charles University in Prague
Malostranske namesti 25, Prague, Czech Republic

marta@ktiml.mff.cuni.cz, kopecky|vojtas@ksi.mff.cuni.cz

Abstract. In this paper we describe the KTIML team approach to *RuleML 2015 Rule-based Recommender Systems* for the *Web of Data Challenge Track*. The task is to estimate the top 5 movies for each user separately in a semantically enriched *MovieLens 1M* dataset. We have three results. Best is a domain specific method like "recommend for all users the same set of movies from Spielberg". Our contributions are domain independent data mining methods tailored for top-k which combine second order logic data aggregations and transformations of metadata, especially 5003 open data attributes and general GAP rules mining methods.
Keywords: rule induction, second order logic rule systems, transformation of metadata to values

1 Introduction and Related Work

Rule-based recommender systems can provide a desirable balance between the quality of the recommendation and the understandability of the explanation for the human user. This challenge targets a new type of recommender problems which uses the Web of data to augment the feature set. The participating systems were requested to find and recommend for each user a limited set of 5 items. The challenge uses a semantically enriched version of the *MovieLens 1M* dataset. Recommender performance is measured by F-measure at top-5 (F@5) and aggregate diversity (ILD). Compactness of explanation is measured qualitatively where good explanations involve small number of easy to understand rules.

Our best result "recommend for all users the same set of movies from Spielberg" is rather specific for this data. Our contribution is top-k focused data mining that combines data aggregations and transformations of metadata, especially 5003 open data attributes. Results are in form of simple SQL queries which can be transformed to second order logic rules. We introduce 2GAP – a second order logic variant of GAP – generalized annotated logic programming rules of [1].

In the area of recommender systems “same-data-challenges” probably the most famous is the *Netflix* challenge [8]. Netflix results were measured by improvement in

RMSE of ranking prediction, while we find F@5 as more challenging metrics. We found relevant mining tricks in [7] and winners of 2014 ACM RecSys Challenge [9].

Going to second order logic was motivated by [2], [4] and [5]. Induction of many valued rule systems was handled in [6] (see [3]). Nevertheless in [6] data sets for induction were much smaller. We hope our method is transferable, nevertheless in this paper we did not check it on other data.

2 Data Preparation, Challenge Understanding

In this chapter we will describe our approach to processing input data.

To be able to handle data and understand them we have chosen to store them and analyze them in the relational database. The python script produced CSV files using data joins with very large results – so a standard 32bit PC or even 64bit desktop PC with usual amount of memory could not process it. Due to the size of the data, practical calculations were done using SQL queries and adapted ML tools. We converted results to simple SQL and constructed corresponding GAP rules.

To be able to store data efficiently, we had to eliminate data redundancy again. Therefore we preprocessed obtained CSV files and loaded data about movies, users and known ratings in form of three database tables. Separately we stored the incidence matrix of movies and 5003 DBPedia attributes.

2.1 Task Analysis, Understanding and Steps of Solution.

After a while spent with playing with the system and sending results in required format (`CSV`, columns `userId,movieId,score`), we discovered that

*the quality of solution **does not** depend on score.*

Further we observed that the recall R^u for user u is computed with respect to a larger candidate set, nevertheless in average of size usually around 10. Hence recall and F-measure are approximately a fixed multiple of precision and for maximizing F@5 it suffices to maximize P@5. We did not try to improve ILD parameter by any activity. We can just observe that results indicate quite high dissimilarity of our top-5 objects.

2.2 Natural Language Extraction of DBPedia Attributes

First, really big obstacle, was the number of movie attributes, especially 5 003 DBPedia binary attributes. First we removed their common prefix[1] and the rest was processed by natural language extraction using database tools. First guess was to take most frequent properties and check their influence of our prediction task. We saw that these do not influence our rule mining achievements on the task significantly.

Our second attempt was based on observation that certain types of properties like Films_directed_by_..., Films_set_in_..., Films_shot_in_..., Films_about_... etc. form

[1] "uri_relation_http://dbpedia.org/resource/Category:"

groups that can be seen as a predicates in form *property=value*. Nevertheless, this did not bring us much further in our rule mining task.

Third attempt brought some progress. We counted occurrences of individual values of properties in all movies and compared it with number of occurrences in rated movies. The bigger the ratio between movie count and rating count the more important property-value pair. The result was submitted as user KSI (seeTable 2). For further use, we aggregate all these properties to a single indicator GoodProperty=1 iff at least one of properties is present.

Our other approach was to create set of explanatory attributes of movies and set them to 1, respectively 0 according to appearance of some word or phrase in the movie flag description. So we created attribute *Spielberg* and set it to 1 for each movie directed, produced or any other way connected to Steven Spielberg, similarly we created attribute *LA* and assigned it to all movies located to LA etc. Then we tried to compare movie ordering based on this attribute with the ordering based of ratings (the ground truth). Attributes producing most corresponding orderings were combined to query that ordered movies using expression**:**

$$100*\text{Movies.Spielberg} + 50*\text{Movies.Original} + \text{Movies.BayesAVG} \quad (1)$$

Where BayesAvg=(3.5*50+AVG_Rating*MovieCNT)/(MovieCNT+50). Shown weights are proportional to number of occurrences of given attribute in movies. On the other hand the ordering is quite robust and is not overly dependent on specific numbers as long as coefficients decrease from left to right. F@5 shown in Table 2 via user SCS_CUNI is surprisingly good. The result can be interpreted that (in ideal case) three quarters of users have at least one Spielberg movie in their top 5. Nevertheless we do not consider this as our challenge contribution, because it is probably domain dependent and rather a property of the respective dataset.

3 Modeling

First, we learned different ML models to predict “interesting” movies. All models gave similarly good results, we aimed for decision tree as our preferred expression language because it can be more easily converted to rules.

The goal was to recommend 5 movies, i.e. to select 5 best (unseen) predictions for any user. It appeared that a tree with more than one hundreds leaves can be pruned to five rules without significant decrease in the goal measure.

Before sending a file for an evaluation, simple post processing was done to transfer the prediction of (any) our model to 5 recommended movies.

3.1 Model Building

As training data, we used records in form (movieID, userID,Age, BayesAVG, GoodProperty, Gender, GenreMatch, TAG), where the TAG=1 iff the movieID was rated by the user, TAG=0 otherwise. We trained several machine learning models. Since there was only 5% positive cases, the equal-cost recommendation was always 0.

If we measure the error as an average of the error on positive and negative cases, we got errors 73% for *Generalized Linear Models*, 72% for *decision trees* and *Naive Bayes* and 52% for *Support Vector Machines*. Since the difference between first three models is small and we have strong preference for rule models we proceed further with the *Decision Tree model* (DT).

3.2 Pruned Model

Our DT model predict preference on movies for users. Our goal was quite different. For a given user predict the top 5 movies. This makes some attribute test superficial (e.g. young users rated more movies, therefore the chance of TAG=1 was higher, but it does not influence the ordering on movies for a given user; similarly, UserAVG and so on). The only user-depended attribute left was the attribute *GenreMatch*, an indicator whether the genre mostly selected by the user is the same as the genre of currently considered movie.

Furthermore, the ordering on most movies was not important since we were interested only in top 5 out of 3.1 thousands. This makes most of the DT splits unimportant. We selected only the 5 most important nodes out of 126 learned (those with the highest percentage of positive cases and a reasonable support). We got F@5 is 0.04978 and precision 0.0714. We hope this dramatic pruning can be successful in other tasks aiming for top-k predictions. These 5 rules are listed in Table 1, all rules have additional condition *GenreMatch*=1.

RulePreference	Rule
0.11	R1:GoodProperty=1
0.25	R2: 113.5<CNT<400
0.29	R3: R1 and R2
0.58	R4: GoodProperty=0& CNT>399
0.57	R5: GoodProperty=1 & CNT>399

Table 1. Rules obtained pruning the decision tree model and weights assigned by relative frequency of the positive examples. „Has good property“ means that at least one property from Table 4 that appears in at least 5 movies has non-zero value, all rules have additional condition GenreMatch=1.

3.3 Post Processing

The overall ordering was given by the combination of the *GenreMatch* (a necessary condition for high ranking), RulePreference as a rough ordering and movie average ranking for ordering movies indifferent for rules. Expressed by a formula:

$$p.Prediction := p.GenreMatch*(p.BayesAVG+100*p.RulePreference) \quad (2)$$

for each user, we ordered all candidate movies according the *Prediction* and selected the first 5 unseen movies.

Query motivated by this approach and the first order one gave Precision: 0.135 (a constant multiple of F@5) and is uploaded as Participant "KTIML", as a main result of a methodology which can be used also in other similar tasks.

Result	Participant	Method	F@5
1	SCS_CUNI	"Spielberg"	0.10681
2	**KTIML**	**Data mining combined with first order**	**0.10085**
3	KSI	Pure first order logic with weighted average	0.05262

Table 2. Result 1 is obtained by domain/data dependent method. Result 2 has to be considered as our challenge contribution.

Although method "Spielberg" achieved better results – this method cannot be used in other application, simply because it is domain and dataset dependent and uses specifically properties of movie data.

4 2GAP – a Rule System Equivalent to Simple SQL Queries

In this chapter we describe the translation of results to a rule system. We use connection between simple SQL queries and logical rules. A query

SELECT $attribute_1$, …, $attribute_n$

FROM $table_1$, …, $table_m$

WHERE conditions

(keeping in mind that `conditions` can contain bounds on variables and some filtering conditions `filter`) is semantically equivalent to the logical rule

$result(attribute_1,\ldots,attribute_n) \leftarrow table_1,\ldots,table_m$, filter

and bounds on variables were applied.

We use (many valued) GAP – Generalized annotated Programs rules of Kifer and Subrahmanian [1]. We interpret truth values as preference degrees. Semantics of these rules is equivalent to database semantics.

A 2GAP rule is a GAP rule in a language extended by atomic predicates corresponding to tables resulting from database aggregations. These predicates are originally defined by second order logic condition equivalent to database aggregation, we assume we have facts in this new atomic predicate filled from database. This can be considered as an alternative approach to that in [2] and [4] (similarly as in [5]).

Assume we generated a table Ordered_Prediction using expression from equation (1). Then SQL query

SELECT UserID, MovieID, 5 FROM Ordered_Prediction WHERE OrdNr <= 5;

corresponds to GAP rule

$SCS\_CUNI\_Movie(u,m):100*x_1+50*x_2+ x_3 \leftarrow$

$\leftarrow$ SPIELBERG(m): x_1 & ORIGINAL(m): x_2 & BAYESAVG(m):x_3

with top-5 semantics. Meaning of this rule is, that whenever (a two valued conjunction)

SPIELBERG(m)$\geq x_1$ & ORIGINAL(m) $\geq x_2$ & BAYESAVG(m) $\geq x_3$ then `SCS_CUNI_Movie(u,m)` $\geq 100*x_1+50*x_2+ x_3$.

The weighted average movie ranking from Table 2user KSI can be depicted by following GAP rule (after transformation of DBPedia attributes this is a first order logic):

KSI_Movie(u, m):(8000*x1+17000*x2+10000*x3+…)*x15 ← MAKEUP(m):x1 & VISUAL(m):x2 & SMIX(m):x3 … NOTRATED(u,m):x15

Recommended movie by data mining can be described (with necessary tuning of weights) by following rule

KTIML_Movie(u.m):(w_1*x_1+w_2*x_2+w_3*x_3)*x_4 ← GENREMATCH(u,m):x_1 & BAYESAVG(m):x_2 & RULEPREFERENCE(u,m):x_3 & NOTRATED(u,m):x_4

Here predicates `GENREMATCH`, `BayesAvg` and `RulePreference` from `Table 1` are atomic 2GAP predicates representable by simple SQL query.

5 Conclusions

We found dataset and task formulation very specific. Unfortunately we have better results for domain specific methods like recommend to all users the same set of movies from Spielberg. Challenge result combine data aggregations, transformations of metadata (especially 5003 open data cloud attributes), selection of relevant attributes and general data mining method. We believe that our method of dramatic pruning can be re-used for other tasks aiming for top-k predictions. All recommendation processing was done in a database engine resulting simple SQL queries were transformed to second order logic GAP rules.

Announcement. Supported by the Czech grants P46 and GACR-P103-15-19877S.

6 References

1. M Kifer, VS Subrahmanian. Theory of generalized annotated logic programming and its applications. Journal of Logic Programming 12,4 (1992) 335-367
2. W. Chen, M. Kifer, D. S. Warren. HILOG: a foundation for higher-order logic programming. Journal of Logic Programming 15,3 (1993) 187 – 230
3. S. Krajci, R. Lencses, P. Vojtas. A comparison of fuzzy and annotated logic programming. Fuzzy Sets and Systems 144,1 (2004) 173-192
4. A. Mohapatra, M. Genesereth, *Aggregates in Datalog under set semantics*, Tech. Rep. 2012
5. Datomic Queries and Rules. http://www.datomic.com/
6. Tomás Horváth, Peter Vojtas. Induction of Fuzzy and Annotated Logic Programs. In ILP 2006, LNCS 4455, 2007, pp 260-274
7. Xavier Amatriain, Josep M. Pujol, Nuria Oliver. I Like It... I Like It Not: Evaluating User Ratings Noise in Recommender Systems. In UMAP '09, Springer 2009, 247 - 258
8. Netflix Prize , http://www.netflixprize.com/,
9. Frederic Guillou, Romaric Gaudel, Jeremie Mary, Philippe Preux. User Engagement as Evaluation: a Ranking or a Regression Problem? ACM 2014, 7-12