Hybrid Recommending Exploiting Multiple DBPedia Language Editions

Ladislav Peska and Peter Vojtas

Faculty of Mathematics and Physics
Charles University in Prague
Malostranske namesti 25, Prague, Czech Republic
peska|vojtas@ksi.mff.cuni.cz

Abstract. In this paper we describe approach of our SemWex1 group to ESWC RecSys Challenge. Our method is based on using Content Boosted Matrix factorization [CBMF], where objects are defined through their content-based features. Features were comprised of both direct DBPedia predicate-object pairs and derived semantical information.
Total of seven DBPedia language editions were used to form our dataset. In the paper we will further describe our methods for semantical information creation, data filtration, algorithm details and settings as well as decisions made during the challenge and dead ends we faced.

Keywords: Hybrid Recommender systems, Linked Open Data, DBPedia, Matrix factorization.

1 Introduction

Recommending and estimating user preferences on the web are both important commerce application and interesting research topic. The amount of data on the web grows continuously and it is impossible to process it directly by a human. Many solutions were adopted ranging from keyword search engines to information aggregators, semantic web or recommender systems. Although majority of the research effort was initially spent on the collaborative filtering based on explicit user rating, collaborative systems might highly suffer from *cold start* problem.

Using attributes of the objects (and hence content-based or hybrid recommender systems) can speed up learning curve and reduce *cold start* (and also *new object*) problem. Various domains and systems however differ greatly in how many and how useful attributes can be provided in machine readable form. This is where Linked Open Data and DBPedia as one of its cornerstones comes into play and with its vast amount of machine-readable data it can be used to populate object attributes and features.

Related Work: Unfortunately it is out of scope of this paper to provide more elaborated overview, so we stick only to the closest work. The preference learning is based on third-party algorithm Content-boosted matrix factorization (CBMF) originally presented by Forbes and Zhu [CBMF]. This method extends classical matrix factorization [Koren] by adding object attributes and stating that each object's

latent factors vector is a function of its attributes latent factors. Our previous experiments with this method on the domain of secondhand bookshop are described in paper [SerSy]. Some decisions made e.g. during dataset preparation are based on observations from this work. The recommending algorithm laid some constraints on usage of RDF triples. The RDF records are mapped directly as attributes of the object. Although some graph based features can be employed, it is not natural for CBMF. We suggest approach of Ostuni et al.[Vito RecSys13] as an interesting approach leveraging graph nature of LOD.

2 Recommending Method

Matrix factorization techniques are currently leading methods for learning user preferences. Given the list of users $U = \{u_1,...,u_n\}$ and objects $O = \{o_1,...,o_m\}$, we can form the user-object rating matrix $\mathbf{R} = [r_{uo}]_{n\times m}$. For a given number of latent factors *f*, matrix factorization aims to decompose original **R** matrix into $\mathbf{UO}^T$ (1), where **U** is $n\times f$ matrix of user latent factors (μ_i^T stands for latent factors vector for particular user u_i) and $\mathbf{O}^T$ is $f\times m$ matrix of object latent factors (σ_i is vector for particular object o_i). Unknown rating for user *i* and object *j* is predicted as $\hat{r}_{ij} = \mu_i^T \sigma_j$. Matrix factorization target is to learn matrixes **U** and **O** minimizing errors on known ratings (usually with some regularization penalty to prevent overfitting. Such equation can be solved e.g. by Stochastic Gradient Descent (SGD) iteratively updating user and object latent factors. See e.g. Koren et al. [5] for more information on matrix factorization techniques.

Content boosted matrix factorization method (CBMF) is based on the assumption that each object's latent factors vector is a function of its attributes. Having $\mathbf{O}_{m\times f}$ matrix of object latent factors, $\mathbf{A}_{m\times a}$ matrix of object attributes and $\mathbf{B}_{a\times f}$ matrix of latent factors for each attribute, the constraint can be formulated as **O = AB**. Under this constraint, we can reformulate both matrix factorization problem (1), its optimization equation and gradient descend equations (2):

$$\mathbf{R} \approx \mathbf{UO}^T = \mathbf{UB}^T\mathbf{A}^T = \underbrace{\begin{bmatrix} \mu_1^T \\ \mu_2^T \\ \vdots \end{bmatrix}}_{n\times f} \times \underbrace{\mathbf{B}^T}_{f\times a} \times \underbrace{\begin{bmatrix} a_1 & a_2 & \dots \end{bmatrix}}_{a\times m} \tag{1}$$

$$\mu_i = \mu_i + \eta\left(\sum_{j\in K_{ui}} (r_{ij} - \mu_i^T \mathbf{B}^T a_j)\mathbf{B}^T a_j - \lambda\mu_i\right) \qquad \sigma_j = \sigma_j + \eta\left(\sum_{(i,j)\in K} (r_{ij} - \mu_i^T \mathbf{B}^T a_j) a_j \mu_i^T - \lambda \mathbf{B}\right. \tag{2}$$

CBMF method has also some drawbacks. One of the most important is time complexity, rising with both number of latent factors *f* and number of attributes *a*. This prevents us from using all crawled attributes and forced us to heuristically choose a sample of them. Also max running time and number of iteration parameters were employed.

The ratings were in some experiments normalized by simple ANOVA model consisting of average rating for user b_u, object b_i and global average μ. Other

additions to this normalization based on DBPedia data were tested, however did not improve evaluation metrics.

$$r_{u,i} = \mu + b_u + b_i + \varepsilon_{u,i} \tag{3}$$

Post-processing was applied for task 3 in order to increase diversity of the resulting set, see Algorithm 1. During evaluation, we have observed high fluctuation of recommended objects, so we have employed bagging over several CBMF runs (object rating was summed over all runs), which highly improved precision with only little diversion penalty.

Algorithm 1: For producing top-k list of recommending objects, k=20, we first produce list of top-l, l > k best rated objects for each user. Then iteratively algorithm take the best object and eliminate all objects with the same author (or any other restricted features). If no more object is available, then the top-l is reset to all objects except already selected ones.

```
function PostProc(List top-l, k, List feat){
   while(selectObj < k){
     reset top-l;
     while(sizeof(top-l)>0){
         get best object from top-l; selecObj++;
         delete O ∈ top-l: feat[O] ∩ feat[bestO] != ⍉;
} } }
```

3 Dataset and Semantics

The dataset used by CBMF consisted of both direct DBPedia triples and added metadata created from them. As for DBPedia, we first downloaded all potentially interesting data and then evolve methods to filter them. The core dataset consisted of RDF triples with patterns: *(_book, ?p, ?o), (?o, ?p, _book), (_author, ?p, ?o) or (?o, ?p, _author)*, where *_book* is book URI and *_author* is an URI of author of the current book. In our approach, we did not distinguish whether *_book* or *_author* is in the role of subject or object in the dataset. For each triple, the corresponding *DBbook_itemID* is also stored. Data were then transformed to fill feature matrix **A** in a following way: Rows consist of *DBbook_itemIDs*, columns consists of all known (seen in the data) combinations of ?o, and ?p and value of each cell is binary information whether for current *DBbook_itemID* exist ?o,?p in the dataset.

Enrichment and Alteration of the Core Dataset: Generally one of disadvantages of using CBMF for LOD data is a flat nature of the object attributes. We can either stick only to the direct attributes, automatically traverse attributes up to a certain depth or we can explore only some parts of DBPedia graph. We choose the third option as using sole direct attributes would result in large loss of information and uniform traversing would on the other hand produce way too much useless data.

Some data alternations were performed when necessary; we mention only more interesting data enrichment and transformation:

Transformation of RDF into **A** matrix is particularly unfriendly to the numeric values, so several meaningful features with numeric values (e.g. number of pages or release date) were mapped into equipotent intervals and further used in that way.

All notions of similar books (e.g. *preceededBy*, *notableWork* etc.) and similar persons (e.g. *influences*, *influencedBy*, *author* etc.) were grouped together and published as *similarWork* and *similarPerson* features.

In some cases the value (RDF object) of a predicate is not so important and the sole existence of the feature may carry enough information. So, for each *?p*, attribute *has_predicate+?p* was added with binary value whether current book has feature *?p*.

We have added information whether the book has wikipage also in other Wikipedia language editions to exploit multilingual nature of users. Also if DBPedia language edition exists, we can add language specific data to the core dataset. Nevertheless the data analysis showed that language editions contain mostly the same information except for *wikiPageWikiLink* property, which was added to the dataset.

Further manually annotated semantical information was added for *authors* and *genres* exploiting axes like serious or fun literature, male or female target audience or clustering genres.

To reflect possible similarity on super categories of books, we also collected 3 levels of super-categories through *skos:broader* property.

Data filtration: The above described raw dataset contained about 2,5M triples, 2800 distinct features and over 400K distinct feature × value pairs which is far beyond reasonable computation time. The algorithm time complexity is dependent on #attributes i.e. feature × value pairs, so we focused on decreasing its number without severe damage to information richness of the dataset. Three basic filters were designed:

- **Feature name filter:** Keep only features not present on the list of useless features. The list was formed manually containing features with no or too little meaning e.g. *dbpedia-owl:wikiPageId*, *rdfs:label*, *rdfs:comment* etc.
- **Feature support filter:** Keep only features with at least k_s support among objects and at least k_{vc} distinct values, with k_s set to 5% and k_{vc} to 2.
- **Feature values support filter:** Keep only feature values, where its support is between k_{v1} and $k_{v2,}$ set to k_{v1} = 5 books and $k_{v2,}$ = 90%.

Setting right boundaries is a bit tricky: basically we need features and values which will reasonably distinguish books into not too small or big groups. The exact setting was tuned experimentally.

After applying filters, we have received a approx. 100 features and 35K distinct feature × value pairs. Although this is already reasonable amount of data for some initial experiments, the algorithm running time was still too slow and only a few iterations could be done. More speculative heuristics were applied thereafter mostly in form of not using / using only certain features or using only top-k feature values according to its support. After series of preliminary evaluations, the resulting dataset was formed after applying:

- Not using super-categories and has_predicate features.
- Use only top-k most supported *similarWork*, *similarPersons* and other feature values (evaluated separately, k=300).

After applying these heuristics, the resulting dataset contains approx. 285K triples, and 60 different features.

4 Results and Discussion

Table 1 contains results of the on-line evaluation. In general were more successful methods with less latent factors or based on smaller datasets. This might be caused by constraint on maximal CBMF running time, or perhaps also because of too much noise within larger datasets. Our future work should definitely include experiments and metrics defining data purity and usefulness specially if comes from third party resources.

Table 1. Results of an online evaluation

Task	Method	Score
Task 1	CBMF (5 lat. factors) + ANOVA	0.9369 RMSE
Task 1	Sole baseline predictors	_ RMSE
Task 2	CBMF (5 lat. factors) + ANOVA	0.555 F-measure
Task 2	CBMF (5 lat. factors)	_ F-measure
Task 3	CBMF (5 lat. factors) + ANOVA	_ F-measure, _ ILD
Task 3	Bagging + post-processing (various CBMF)	_ F-measure, _ ILD

During work on the challenge we have discovered several dead ends and problems, namely super-categories are often too general to provide any reasonable information, hypothesis about importance of feature occurrence itself (*has_predicate* feature) was also not confirmed. The effect of using multiple DBPedia language editions is questionable, however we do not abandon this idea yet as the challenge dataset seems to be comprised mostly from English-speaking users.

On the other hand ANOVA normalization was very useful in both tasks 1 and 2 and other variants of rating normalization should be examined. The post-processing effectively increased diversity and bagging improved precision with minimal decrease of diversity for task 3, so we encourage others to use it as well.

Acknowledgments: This work was supported by grants SVV-2014-260100, P46 and GAUK-126313.

References

1. Forbes, P. & Zhu, M. Content-boosted matrix factorization for recommender systems: experiments with recipe recommendation. *RecSys 2011, ACM,* **2011**, 261-264
2. Koren, Y.; Bell, R. & Volinsky, C. Matrix Factorization Techniques for Recommender Systems. *Computer, IEEE,* **2009**, *42*, 30-37
3. Ostuni, V. C.; Di Noia, T.; Di Sciascio, E. & Mirizzi, R. Top-N recommendations from implicit feedback leveraging linked open data, *RecSys 2013, ACM,* **2013**, 85-92
4. Peska, L.; Vojtas, P.: Using LOD to Improve Recommending on E-Commerce. *To appear on SerSy 2013, http:// www.ksi.mff.cuni.cz/~peska/sersy13.pdf*