Using Linked Open Data to Improve Recommending on E-Commerce

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

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

ABSTRACT
In this paper, we present our work in progress on using LOD data to enhance recommending on existing e-commerce sites. We imagine a situation of e-commerce website employing content-based or hybrid recommendation. Such recommending algorithms need relevant object attributes to produce useful recommendations. However, on some domains, usable attributes may be difficult to fill in manually and yet accessible from LOD cloud.

For our pilot study, we selected the domain of secondhand bookshops, where recommending is extraordinary difficult because of high ratio of objects/users, lack of significant attributes and small number of the same items in stock. Those difficulties prevents us from successfully apply both collaborative and common content based recommenders. We have queried Czech language mutation of DBPedia in order to receive additional information about objects (books) and use them as Boolean attributes for hybrid matrix factorization method. Our approach is general and can be applied on other domains as well.

Proposed methods were successfully tested in an off-line experiment; however we needed to cope with several technical difficulties and obstructions described in the paper, which may hinder widespread of such approaches.

Categories and Subject Descriptors
H.3.3 [**Information Systems**]: Information Search and Retrieval - *Information Filtering*

General Terms
Measurement, Human Factors, Experimentation.

Keywords
Hybrid recommender systems, Linked Open Data, DBPedia, content-based attributes, e-commerce, matrix factorization.

1. INTRODUCTION
Recommending on the web is both an important commercial application and popular research topic. The amount of data on the web grows continuously and it is virtually impossible to process it directly by a human. The keyword search engines were adopted to fight information overload but despite their undoubted successes, they have certain limitations. Recommender systems can complement onsite search engines especially when user does not know exactly what he/she wants.

Many recommender systems, algorithms or methods have been presented so far. Initially, the majority of research effort was spent on the collaborative systems and explicit user feedback. Collaborative recommender systems suffer from three well known problems: *cold start*, *new object* and *new user problem*.

New user / *object* problem is a situation, where recommending algorithm is incapable of making relevant prediction because of insufficient feedback about current user / object. The *cold start* problem refers to a situation short after deployment of recommender system, where it cannot relevantly predict because it has insufficient data generally.

The new object problem became even more important on domains with high object fluctuation (e.g. fast aging objects like news articles or limited amount of items per object type). Using attributes of objects and hence content based or hybrid recommender systems can speed up learning curve and reduce both cold start and new object problems. Various domains however differ greatly in how many and how useful attributes can be provided in machine readable form. However we can use some of the LOD datasets to enhance our object's attributes and thus improve recommendation quality.

1.1 Our motivation
Despite the widespread of recommending systems, there are still domains, where creating useful recommendations is very difficult.

- Auction servers or used goods shops have often only one object of given type available which prevents us from applying collaborative filtering directly.
- For some domains e.g. films, news articles or books it is difficult to define and maintain all important attributes hindering content based methods.
- Websites with relatively high ratio between #objects / #users will generally suffer more from cold start and new user problems.

For our study, we chose secondhand bookshops domain which includes all above mentioned difficulties. The main problem of the book domain is that similarity between books is often hidden in vast number of attributes (characters appeared, location, art form or period, similarity of authors, writing form etc.). Although those attributes are difficult to design and fill, it is not impossible. But in the most cases, only one book is available in the secondhand bookshop, so creating a new record must be fast and simple enough that potential purchase could eventually cover the

SeRSy'13, October 13, 2013, Hong Kong, China.

costs of work. Collaborative recommender systems can be used, but their efficiency is hindered both by high ratio between *#objects/#users* and the fact that each object can be purchased only once.

On the other hand, Wikipedia covers the book domain quite well, so our main research question is whether we can effectively use information available on Linked Open Data cloud (e.g. DBPedia) to improve recommendation on difficult domains such as secondhand book shop.

1.2 Main contribution

The main contributions of this paper are:

- Proposing system to on-line querying Linked Open Data to enrich object's content information
- Experiments with real-world e-commerce data evaluating added value of mined attributes.
- Identifying key problems hindering development of similar systems.

The rest of the paper is organized as follows: review of some related work is in section 2. In section 3 we how to incorporate LOD into recommending systems in general, section 4 describes our selection of recommending algorithms suitable for the task and section 5 discuss specific needs of secondhand-bookshop domain. Section 6 discusses experiments held on Czech secondhand bookshop and finally section 7 concludes our paper and points to some future work.

2. RELATED WORK

Although areas of recommender systems and Linked Open Data have been extensively studied recently, papers involving both topics are not very common. We suggest Konstan and Riedl [7] paper as a good overview for recommender systems domain and Bizen, Heath and Berners-Lee [1] for LOD.

The closest to our work is the research by Di Noia et al. [2], whose aim was to develop content based recommender system for a movie domain based sole on (multiple) LOD datasets. Similarly A. Passant [9] developed *dbRec* – the music recommender system based on DBPedia dataset. Both authors have however different initial point of view: they struggle to develop system based sole on LOD datasets. Although Di Noia et al. used MovieLens 1M dataset in their experiments, they did not claim it to be an integral part of their recommender system, only a testing platform.

Our approach is different in several ways: First we focus on e-commerce domain with its rather unique requirements and possibilities. We expect to have running e-commerce portal with its objects and eventually also existing recommender system. Hence LOD datasets are used only as means of recommendation improvements.

This scenario might looks similar as Di Noia et al. experiments with MovieLens, but we stressed on capability of the proposed recommender system to cope with and to recommend for objects without corresponding LOD data. Although the technique of integrating LOD and non-LOD data is similar in both Di Noia et al. and our approach, recommending procedure differs due to this assumption.

On the contrary of A. Passant [9], who used LOD to create both recommender system and user interface, we need to cope with inaccurate and missing interface between LOD and non-LOD datasets and missing objects. This fact affects choice of recommending (or similarity) method as we can either separate LOD-based and non-LOD recommendations and use e.g. SimRank[6] for LOD or (our choice) integrate LOD data into the objects structure and use e.g. content aware matrix factorization[3]. Using matrix factorization also allows us to be less strict in validation and evaluation of LOD input.

Among other work connecting areas of recommender systems and LOD we would like to mention paper by Heitmann and Hayes [5] using Linked Data to cope with new-user, new-item and sparsity problems and Goldbeck and Hendler's work on FilmTrust [4]: movie recommendations enhanced by FOAF trust information from semantic social network.

Matrix factorization techniques [8] are currently main-stream algorithm to learn user preferences gaining their popularity during NetFlix prize. There are also several implementation of hybrid matrix factorization using taking into account object's attributes. We use *Content-boosted matrix factorization* as proposed in Forbes and Zhu [3] in our experiments.

This paper follows on our preliminary work [10], we have improved data mining procedure in order to increase objects coverage, used hybrid recommending method instead of pure content-based to improve recommending quality and run large scale experiments.

3. ENHANCING RECOMMENDER SYSTEMS WITH LINKED OPEN DATA

In this section, we will briefly describe our architecture to collect LOD and their usage in recommender system. The architecture itself is rather simple and straightforward, but we believe that this chapter can be interesting from the software engineering point of view as we try to point out several issues, which should be bore in mind.

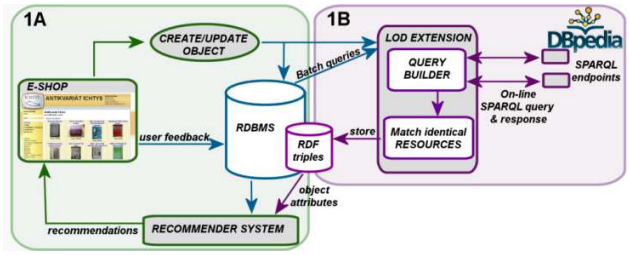

Figure 1: Top-Level architecture of the system: 1A is original e-commerce system with a content-based or hybrid recommender; 1B is its extension for querying and storing LOD datasets.

The top-level architecture is shown on Figure 1. System maintains one or more SPARQL endpoints to various LOD datasets. The connections are usually REST APIs or simple HTTP services. Whenever an object of the system is created or edited, the system will automatically query each SPARQL connection with best possible object specification.

The important step while using more than one LOD dataset is matching identical resources. The difficulty of this step is however highly dependent on the datasets. Resources are then stored in the system triples store (relational database table was sufficient during our experiments) or alternatively mapped into the object-attribute structure immediately.

Each object is also queried periodically via batch queries as the LOD datasets can provide more information over time. We have also considered using local copies of LOD datasets. This approach would however result in excessive burden to both data storage and system maintenance and also prevent us from using up-to-date dataset.

3.1 Using LOD in Recommender System

Unless we want to use specific graph-based similarity measures e.g. SimRank [6] (which are appropriate for LOD, but would cause problems for objects without matching LOD data) we need to map mined RDF into the Object-Attribute structure. In our previous work, we used following mapping:

- *RDF Subject* ≈ recommendable object (e.g. product of an e-shop)
- *RDF Predicate* ≈ Attribute name
- Set of *RDF Objects* (with the same Subject and Predicate) ≈ Attribute value

Set attributes were thereafter compared with Jaccard similarity, original attributes of an object remains the same compared with appropriate similarity metrics. Although such mapping achieved quite positive results, it has one important weakness: various *RDF Objects* of the same *Predicate* may have highly different information value. Good example of this phenomenon is *dbPedia:wikiPageWikiLink Predicate* (see Section 5 for more details). Due to this, we decided to use other approach:

- *RDF Subject* ≈ recommendable object
- <*RDF Predicate* × *RDF Objects*> ≈ Binary attribute

Original object attributes should be transformed into binary as well. This setting leads directly to some form of matrix factorization as the matrix itself is typically very large and yet sparse enough (approx. 70000 attributes, 5000 objects and density ≈ 0.1% during our experiments). One advantage is that significance of each *RDF Object* is considered separately, however number of attributes can grow unbound unacceptably rising time demands. To improve this problem, we proposed two methods to reduce number of attributes in Section 4.2.

4. RECOMMENDING ALGORITHMS

Matrix factorization techniques are currently leading methods for learning user preferences. We choose to use Forbes and Zhu's C*ontent boosted matrix factorization method* [2] as it incorporates object attributes directly into the matrix factorization. We will briefly describe their approach and its (dis)advantages. For space reasons we skip more elaborated introduction on matrix factorization. We suggest consulting Koren et al. [5] instead.

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 $\mathbf{R} = [r_{uo}]_{n\times m}$. With lack of explicit feedback, user-object rating r_{uo} in our case is defined as Boolean information whether user *u* visited object *o*. We will keep using term "*rating*" for this implicit information throughout the rest of the paper as it is more convenient in other Matrix Factorization related literature.

For a given number of latent factors *f*, matrix factorization aims to decompose original **R** matrix into $\mathbf{UO}^T$, 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 of latent factors for particular object o_i).

$$\mathbf{R} \approx \mathbf{UO}^T = \underbrace{\begin{bmatrix} \mu_1^T \\ \mu_2^T \\ \vdots \end{bmatrix}}_{n\times f} \times \underbrace{[\sigma_1 \quad \sigma_2 \quad \ldots]}_{f\times m} \tag{1}$$

Unknown rating for user *i* and object *j* is predicted as $\hat{r}_{ij} = \mu_i^T \sigma_j$. Our target is to learn matrixes **U** and **O** minimizing errors on known ratings. Regularization penalty is added to prevent overfitting altogether forming optimization equation:

$$\min_{\mathbf{U},\mathbf{O}} \left\| \mathbf{R} - \mathbf{UO}^T \right\|^2 + \lambda(\left\| \mathbf{U} \right\|^2 + \left\| \mathbf{O} \right\|^2) \tag{2}$$

This equation can be solved e.g. by Stochastic Gradient Descent (SGD) technique iterating for each object and user vectors

$$\begin{aligned} \mu_i &= \mu_i + \eta \Big(\sum_{j\in K_{ui}} (r_{ij} - \mu_i^T \sigma_j)\sigma_j - \lambda \mu_i \Big) \\ \sigma_j &= \sigma_j + \eta \Big(\sum_{i\in K_{oj}} (r_{ij} - \mu_i^T \sigma_j)\mu_i - \lambda \sigma_j \Big) \end{aligned} \tag{3}$$

Where η is learning rate, K_{ui} set of all objects rated by user u_i and K_{oj} set of all users, who rates object o_j. We use this method as one of our baselines.

4.1 Content boosted matrix factorization

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 $\mathbf{A}_{m\times a}$ matrix of object attributes and $\mathbf{B}_{a\times f}$ matrix of latent factors for each attribute, the constraint can be formulated as:

$$\mathbf{O} = \mathbf{AB} \tag{4), (6 [3])}$$

Under the constraint (4), we can reformulate both matrix factorization problem (1), its optimization equation (2) and gradient descend equations (3):

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

$$\begin{aligned} \mu_i &= \mu_i + \eta \Big(\sum_{j\in K_{ui}} (r_{ij} - \mu_i^T \mathbf{B}^T a_j)\mathbf{B}^T a_j - \lambda \mu_i \Big) \\ \sigma_j &= \sigma_j + \eta \Big(\sum_{(i,j)\in K} (r_{ij} - \mu_i^T \mathbf{B}^T a_j) a_j \mu_i^T - \lambda \mathbf{B} \Big) \end{aligned} \tag{3a}$$

(8,9 [3])

4.2 Reducing number of attributes

Both Forbes and Zhu's and our own experiments corroborated that *CBMF* method improves recommendation quality in both RMSE, nDCG, Average Position or Presence at top-k metrics. Price for the improvement is time complexity which rises with number of attributes a_i. Forbes and Zhu's experiments were made on recipes domain with 7000 attributes representing recipe's ingredients, ours on travel agency dataset with approx. 1000 attributes of each tour. In both cases the number of attributes was relatively stable and can be internally controlled. While employing LOD, we start to lose control on number of attributes. In our presented (still rather small) experiment, we have mined

approx. 70000 different RDF objects for 5000 books, which greatly affects computation time. In order to reduce number of attributes, we defined and tested two heuristics:

Reducing based on attribute frequency keeps only top-k (Boolean) attributes with highest frequency among set of objects. Method is based on assumption that infrequent attributes represents marginal properties useless to determine objects similarity. Opposite metric is possible too to reveal DBPedia equivalents of stop-words, but we did not find any too frequent attributes in our dataset.

Reducing based on latent factors first computes *CBMF* with small number of latent factors, iterations and/or for subset of data and then select top-k attributes with highest contribution to the latent factors (a_i with maximal sum of their attributes latent factors $\sum_{j=1}^{f}\mathbf{B}_{i,j}$ from equation 1a). The idea of this method is to use only attributes which is enough involved in composing hidden (latent) user and object factors – see e.g. Fig. 2 in Koren et al. [8]. This method is however less adaptable to the new objects or new features than the previous one due to overall more complicated *insert new row* operation for the most matrix factorization methods. This fact should be considered while deploying on real-world system as it may cause inaccuracies on domains with high object fluctuation.

5. USING LINKED OPEN DATA IN SECONDHAND BOOKSHOP DOMAIN

In our previous work [10] we explored LOD cloud in order to find usable dataset for Czech secondhand bookshop. Given the language constraint (book titles in Czech) and need for suitable interface (SPARQL endpoint), we found only one available dataset: Czech version of DBPedia[1]. There is unfortunately no common unique identifier within the books domain. Using ISBN is possible, but it identifies each issue, publisher or language version separately and thus will lead to information loss.

We used combination of author and book title keyword search and correct rdf:type in our previous work, however another problem arose: Czech Wikipedia contains relatively large amount of books and writers, but the vast majority of them have no infobox attached and thus cannot be properly identified. Yet they still contain information about Wikipedia pages linked to them[2], which can be valuable in recommendation. In order to use also these resources we need to rollback to simple author and book keyword search. This may lead to substantial inconsistencies due to absence of names disambiguation; however we take advantage of choice of recommending method. As contribution of each attribute (resource) is determined separately to minimize difference between original and factorized matrixes $\left\| \boldsymbol{R} - \boldsymbol{U}\boldsymbol{O}^T \right\|$, irrelevant attributes received by incorrect data mining should be suppressed automatically. Figure 2 shows example of SPARQL query used and portion of its result.

With given settings, we were able to collect additional data about 4980 out of 9500 books (52% coverage instead of 7.3% made by more restrictive mining in our previous work). Altogether over 64000 different attributes were mined, but 34000 of them were present only on one object and were discarded immediately. One object has in average 60 non-zero attributes.

[1] http://cs.dbpedia.org

[2] See e.g. http://cs.dbpedia.org/page/R.U.R.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

select ?p, ?o
where {
 ?book rdfs:label ?book_name .
 ?book_name bif:contains '( "Návrat" and "krále")'.

 ?book ?p ?o.
}
```

p	o
rdf:type	dbpedia:WrittenWork
rdf:type	schema.org/Book
rdfs:label	"Návrat krále"@cs
dbpedia:numberOfPages	409
dbpedia:sourceCountry	csdbpedia:Spojene_kralovstvi
dbpedia:author	csdbpedia:J._R._R._Tolkien
dbpedia:previousWork	csdbpedia:Dve_veze
dbpedia:wikiPageWikiLink	csdbpedia:John_Ronald_Reuel_Tolkien
dbpedia:wikiPageWikiLink	csdbpedia:Sauron

Figure 2: Example of SPARQL query about The Return of the King (in Czech "Navrat krale") and a portion of returned data.

6. EXPERIMENTS

In order to prove our theory as feasible, we performed a series of off-line experiments on real users of a Czech secondhand bookshop. The book shop itself also contains some attributes useful for recommendation: *book title*, *author name*, *book category* and *book price*. For the purpose of recommendation, nominal attributes were transformed into set of Boolean attributes one attribute per value and *book price* was separated into 10 equipotent interval. Again attributes with only one object was discarded resulting into 820 new attributes.

6.1 Recommending methods

In order to test both benefits of using LOD data and methods to reduce number of attributes, we have implemented following methods:

- ***Baseline****:* standard SGD matrix factorization.
- ***Attributes****:CBMF* method using only original book shop attributes.
- ***LOD****: CBMF* using only attributes mined from LOD.
- ***LOD+Attr****: CBMF* using both original book shop attributes and attributes mined from LOD.
- ***LOD+Attr(freq. reduced)****: CBMF* using only top-1000 attributes determined by attribute frequency.
- ***LOD+Attr(lat. fact. reduced)****:CBMF* using only top-1000 attributes determined by latent factors contribution.

All methods were initialized with 10 latent factors and 1 hour maximal computation time. We plan to experiment with other method settings in our future work.

6.2 Experimental goals and success metrics

Before the experiment we need to set experiment goals and success metrics. As the datasets contains only implicit feedback, we cannot rely on user rating and related error metrics e.g. RMSE or MAE (no need to mention that those metrics do not reflect well real-world success metrics anyway). The Precision / Recall methods are also problematical as we can obtain only positive

implicit feedback (user visited object) or its absence which however cannot be automatically interpreted as negative feedback (user might not be aware of the object). Typical usage of recommender systems in e-commerce is to present list of top-k objects to the user. We let recommending methods to rank objects and denote as success if the algorithm manages to rank well enough those objects, we have some evidence of their positive preference.

As we lack any explicit feedback, we need to infer positive preference from the implicit data. For the purpose of this rather early work we consider page-view action (Boolean information whether user *u* visited object *o*) as an expression of positive user preference. More formally, relevance $r_{u,o}$ of object *o* for user *u* is defined as:

$$r_{u,o} = \begin{matrix} 1\ IFF\ u\ \text{visited}\ o \\ 0\ OTHERWISE \end{matrix} \qquad (4)$$

It is possible to use more selective meanings of positive preference e.g. to consider only purchased objects as positively preferred or require more feedback to confirm his/her preference. However this will lead to insufficient data in the test set so we leave the problem of finer grained preference to the future work.

We adopt normalized distributed cumulative gain (***nDCG***) as a standard metrics to rate relevance of list of objects. The premise of nDCG is that relevant documents appeared low in the recommended list should be penalized (logarithmical penalty applied) as they are less likely to attract user attention. This fits well into the recommending scenario, where lower-ranked objects are presented to the user on less desirable positions. It is also possible to restrict DCG to sum only up to *top-k*th position as usually only *top-k* objects are shown to the user. However there is no justification to set any particular *top-k* and the list of eligable objects for recommendation could be pre-filtered (e.g. keep only objects from certain category if user is browsing the category) so objects on lower ranks keeps some value too.

Results of ***Presence@top-k*** and ***Average position*** metrics are shown as they have more intuitive connection to the data. ***Presence@top-k*** for *arbitrary fixed user* and recommending method is defined as sum of $r_{u,o}$ of *top-**k*** best objects according to recommending method for current user. ***Presence@top-k*** is then summed over all users. ***Average position*** is defined as average of positions of objects with $r_{u,o}$=1 in recommending list (defined by current recommending method successively for all users).

6.3 Evaluation procedure

Recommending method evaluation was carried out as follows: For each user, his/her click stream was divided into two halves according to its timestamp – earlier data serving as train set and following as test set. Note that only users with at least two visited objects qualify for the experiment. There are other ways to divide train/test set e.g. to apply cross-validation, but we rather took advantage of possibility to use and compare stream or time-aware algorithms on the same dataset in future. The resulting train set contains 4025 records from 3049 users. Test set contains 4725 records.

Then for each user, each method rate all objects, sort them according to the rating and look up positions of objects from the test set and compute ***nDCG*** and ***P@top-k***. In production recommender system, we should take into account also other metrics like diversity, novelty or serendipity and probably want to pre-select list of candidate objects, but for purpose of our experiment, we will focus on rating only.

6.4 Experiment results

Figure 3 displays results of recommending methods in ***nDCG*** aggregated by the train set size and Figure 4 shows distribution of ***P@top-k*** up to top-150. Although smaller top-k would be used in the real deployment, the list of objects eligible for recommendation would be probably pre-filtered too, leaving some influence also to objects beyond typical top-k boundary.

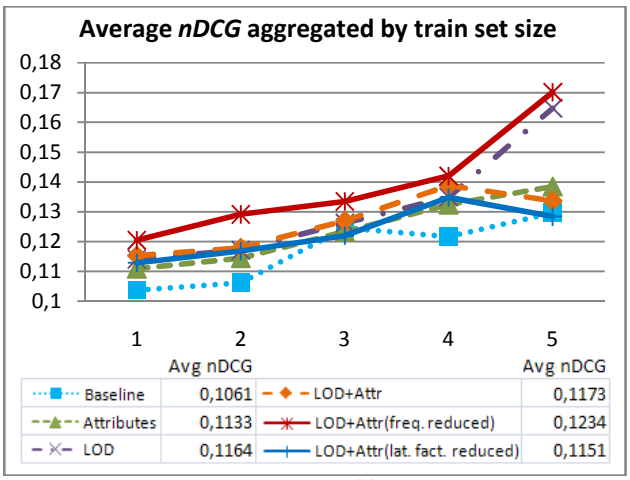

Figure 3: Average nDCG aggregated by train set sizes per user. Legend shows average nDCG per all users and train set sizes.

LOD+Attr (freq. reduced) method achieved the best results in *nDCG* and is statistically significantly better than any other method (p-value<10^{-4}), but *LOD+Attr (lat .fact. reduced)* outperformed other methods in average position (p-value<0.002). All methods are stat. sign. better than *Baseline* (p-value<10^{-6}) and *LOD+Attr* slightly outperforms *Attributes* (p-value<0.03).

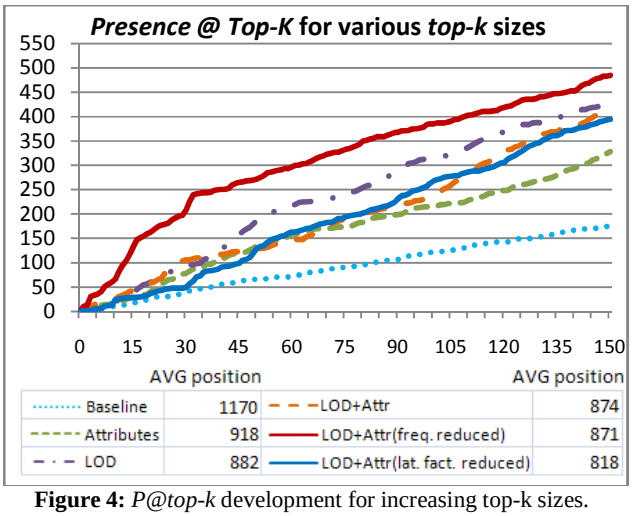

Figure 4: *P@top-k* development for increasing top-k sizes. Legend shows average position of preferred objects.

We can conclude that using additional data from LOD datasets can significantly improve recommendation quality (even in situation when received data are messy and possibly incorrect) if proper recommending method is chosen. Reducing number of attributes is promising approach to both speed up computation and improve results. Both proposed reduction methods looks promising and we will continue to experiment with them and/or combine them together, although reduction based on latent factors might suffer from time-consuming updates and thus should be

deployed carefully, only if more significant improvements are shown.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we aim to improve recommendations on problematical e-commerce systems by using additional data collected from Linked Open Data Cloud. We showed several difficulties which current recommender systems may encounter and a domain (secondhand bookshops) where all those difficulties can be found.

A simple method for enhancing objects with LOD data was presented and possibilities of its utilization were discussed. We chose to use them as Boolean attributes and select hybrid matrix factorization method to derive top-k recommended objects.

The off-line experiments held on the real visitors of Czech secondhand bookshop corroborates our assumption, that enhancing recommender systems with LOD data can improve recommendation quality and we managed to improve one of the main drawbacks of our previous work – low object coverage.

Future work involves e.g. experimenting with other recommendation methods or parameters of current ones. We would like to also consider approaches to enhance matrix factorization with other implicit feedback and also and improving Czech DBPedia mapping rules in order to receive more precise data. The possibility of automatic translation of book names should be also considered in order to use also non-Czech resources.

8. ACKNOWLEDGMENTS

The work on this paper was supported by the grant SVV-2013-267312, GAUK-126313 and P46.

REFERENCES

[1] Bizer, C.; Heath, T. & Berners-Lee, T. Linked Data - The Story So Far. *Int. J. Semantic Web Inf. Syst.*, **2009**, *5*, 1-22

[2] Di Noia, T.; Mirizzi, R.; Ostuni, V. C.; Romito, D. & Zanker, M. Linked open data to support content-based recommender systems. *In I-SEMANTICS '12, ACM*, **2012**, 1-8

[3] Forbes, P. & Zhu, M. Content-boosted matrix factorization for recommender systems: experiments with recipe recommendation. *In RecSys 2011, ACM*, **2011**, 261-264

[4] Golbeck, J. & Hendler, J. FilmTrust: movie recommendations using trust in web-based social network. *In CCNC 2006, IEEE*, **2006**, *vol. 1*, 282-286

[5] Heitmann, B. & Hayes, C. Using Linked Data to Build Open, Collaborative Recommender Systems. *AAAI Spring Symposium: Linked Data Meets Artificial Intelligence*, **2010**

[6] Jeh, G. & Widom, J. SimRank: a measure of structural-context similarity. *In KDD '02*, ACM, **2002**, 538-543

[7] Konstan, J. & Riedl, J. Recommender systems: from algorithms to user experience. *UMUAI*, 2012, 22, 101-123

[8] Koren, Y.; Bell, R. & Volinsky, C. Matrix Factorization Techniques for Recommender Systems. *Computer, IEEE Computer Society Press*, **2009**, *42*, 30-37

[9] Passant, A. dbrec - Music Recommendations Using DBpedia *In ISWC 2010, Springer, LNCS*, **2010**, 209-224

[10] Peska, L.; Vojtas, P.: Enhancing Recommender Systems with Linked Open Data. *In FQAS 2013, Springer, LNCS*, **2013**, 483-494

While e-commerce sites can be extremely convenient, they may come up short in certain areas. Most notable is the inability to provide the same personal customer service you would find in a brick-and-mortar store. If you want to surpass the competition, though, your business can still find ways to improve online customer service. The value of customer service. Whether you realize it or not, customer service plays a major role in most of the purchases you make.Â When you're looking for ways to improve your e-commerce site, analyze your customer service and look for areas where you can improve. Image Credit: naveebird / Getty Images. business.com editorial staff. Record ecommerce competitionâ€”fueled by legacy wholesalers, global retail giants, and product categories not traditionally purchased onlineâ€”is driving up customer acquisition costs. Many new competitors are not equipped to compete on customer experience, a top differentiator online, giving an edge to brands with immersive omnichannel experiences. Shoppers worldwide fuel ecommerce supercycle. At the height of the COVID-19 pandemic, 10 years of ecommerce growth happened in just 90 days. A recommendation engine filters the data using different algorithms and recommends the most relevant items to users. It first captures the past behavior of a customer and based on that, recommends products which the users might be likely to buy. If a completely new user visits an e-commerce site, that site will not have any past history of that user. So how does the site go about recommending products to the user in such a scenario? One possible solution could be to recommend the best selling products, i.e. the products which are high in demand.Â To improve on this type of system, we need an algorithm that can recommend items not just based on the content, but the behavior of users as well. 2.3.2 Collaborative filtering. Let us understand this with an example.