# Exploring Semantic Features for Producing Top-N Recommendation Lists from Binary User Feedback

Nicholas Ampazis and Theodoros Emmanouilidis

Intelligent Data Exploration and Analysis Laboratory (IDEAL),
Department of Financial and Management Engineering,
University of the Aegean,
82100, Chios, Greece
{n.ampazis,emman}@fme.aegean.gr
http://labs.fme.aegean.gr/ideal

**Abstract.** In this paper, we report the experiments that we conducted for two of the tasks of the ESWC'14 Challenge on Linked Open Data (LOD)-enabled Recommender Systems. Task 2 and Task 3 dealt with the top-N recommendation problem from a binary user feedback dataset and results were evaluated on the accuracy and diversity respectively of the recommendations produced in a Top-N recommendation list for each user. The DBbook dataset was used in both tracks in which the books had been mapped to their corresponding DBpedia URIs. Since the mappings could be used to extract semantic features from DBpedia, in all our experiments, we avoided the use of any collaborative filtering methods (e.g. user/item K-nearest neighbors and matrix factorization approaches) and instead focused exclusively on the semantic features of the items. Even though the performance of our methods did not beat the best performing approaches of other teams, our results indicate that it is indeed feasible to create effective recommender systems which fully understand the items they deal with by utilizing information from the Semantic Web.

**Keywords:** Top-N Recommendations, Content-based Recommender Systems, Semantic Web

## 1 Introduction

The literature review of studies on recommender systems (see for example, the Netflix prize solutions summarizing paper of "Bellkor's Pragmatic Chaos" team [1,2]), readily makes apparent that latent factor models (matrix factorization, singular value decomposition, etc), and K-nearest neighbors approaches are among the ones most suited to the problem. However, besides the knowledge of explicit user-item ratings, additional information (if available) can be exploited to define or improve similarities between users and/or items.

Recent developments in the Semantic Web community now enable us to set up links between items in different data sources, and allow us to automatically extract content and meta information for the available items. Item similarity can hence be calculated based on the content comparison of the given items. In the literature this

approach is known as content-based recommendation [3]. Unlike collaborative filtering approaches, in a content-based recommender system, recommendations are based solely on the profile built up by analyzing the content of items that the target user has interacted with in the past. The recommendation problem hence becomes a search for items, the content of which is most similar to the content of items already preferred by the target user. In addition, users can be modeled on the basis of the items that they have collected and thus we can define content-based similarity metrics between the users and the different items.

In this work, we explored a content-based recommendation approach for both Tasks 2 and 3 of the ESWC'14 Challenge on LOD-enabled Recommender Systems and we report on the obtained results. By using LOD and semantic technologies alone, in favour of adopting any of the well known collaborative filtering approaches, we developed a knowledge-enabled content-based recommendation methodology which performed sufficiently well on both tasks. This indicates that content-based approaches that rely entirely on the utilization of semantic features can perform reasonably well on a variety of recommendation metrics.

## 2 Methodology

Both tracks utilized the Dbook dataset which contains user preferences on items retrieved from the Web. The common training set provided for Task 2 and Task 3 contained tab-separated triplets of the form userID/itemID/rating. The ratings were binary, with 1 indicating that the item was relevant for the user, and 0 meaning irrelevant. The training set contained 72372 ratings that 6181 users gave to 6733 items. Overall 8170 book titles were available in the Dbook dataset which were mapped to their corresponding DBpedia URIs. Those DBpedia URIs were used to extract semantic features from DBpedia by issuing SPARQL queries at the endpoint `http://dbpedia.org/sparql`. For each $DBpedia_uri we extracted the subjects and author(s) using the following queries respectively:

SELECT ?o WHERE { <"$DBpedia_uri"> <http://purl.org/dc/terms/subject> ?o.}

SELECT ?o WHERE { <"$DBpedia_uri"> <http://dbpedia.org/ontology/author> ?o.}

We extracted 7079 distinct subjects and 3046 different authors which were used to represent documents as binary semantic feature vectors in $\mathbb{R}^N$ with $N = 7079+3046 = 10125$.

For running our experiments we utilized a PostgreSQL[1] database schema which is shown in Figure 1. The "ratings" table contains the binary ratings training set and the table "features_documents" contains the binary document feature vectors described above. For representing the document vectors in the database we employed MADlib[2] which is an open-source library for scalable in-database analytics with support for PostgreSQL. Madlib implements a sparse vector data type, named "svec", which provides compressed storage of vectors that have many duplicate elements (in our case zeros). The svec type employs a simple Run Length Encoding (RLE) scheme to represent sparse vectors as pairs of count-value arrays. For example, the svec array representation:

'{1085,1,3777,1,532,1,1682}:{0.0,1.0,0.0,1.0,0.0,1.0,0.0}'::madlib.svec

[1] http://www.postgresql.org
[2] http://madlib.net

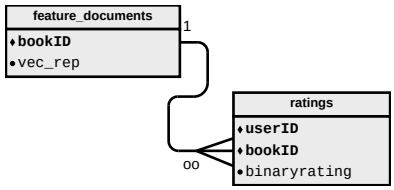


**Fig. 1.** Database schema

says that 1085 occurrences of 0 are followed by 1 occurence of 1, followed by 3777 occurrences of 0, etc. This example uses just 7 integers and 7 floating point numbers to store the array. Further, it is easy to implement vector operations that can take advantage of the RLE representation to make computations faster. The SVEC module provides a library of such functions like adding svec vectors and calcuating distances between them. The field "vec_rep" in the "features_documents" contains the svec vector representation of each bookID.

Our approach for providing Top-N recommendation is that users can be modeled on the basis of the items for which they've expressed a preference. Thus users can be also represented as vectors in $\mathbb{R}^N$ ($N = 10125$), where their feature vectors can be calculated iteratively by adding or subtracting the document semantic feature vectors for which they've expressed a positive or negative signal respectively. This allows us to embed the users onto the document semantic feature space and define content-based similarity metrics between the users and the different items. Consequently we are able to populate the Top-N recommendation list for each user with the items that exhibit the highest similarities with the user's feature vector.

The first listing in Algorithm 1 depicts the MADLib SQL methodology for costructing the feature vector "uvec" for each $DBbook_userID in the training set. For each user we loop over the items for which this user has expressed a binary rating and add or subtract the correspond item feature vector according to the rating being 1 or 0 respectively. At the end of looping process each user has been assigned a semantic feature vector which can then be used to provide Top-N recommendations for Tasks 2 and 3 as explained below.

## 3 Task 2 experiments

Task 2 participants were asked to add a corresponding relevance score for specific user-item pairs in the evaluation dataset. The relevance scores were used by the evaluation service to form a Top-5 item recommendation list for each user. Thus for each user only items in the evaluation set were considered to form the Top-5 recommendation list. The evaluation metric for the task was the F-measure@5[3].

Listing 2 in Algorithm 1 shows how the relevance score for each "$DBbook_userID / $DBbook_itemID" pair in the evaluation can be calculated within the database, provided that the semantic feature vector for the user in question has been calculated as in

[3] http://sisinflab.poliba.it/semanticweb/lod/recsys/2014challenge/eswc2014-lodrecsys-metrics_evaluationservice.pdf

**Algorithm 1** Methodology in SQL

```
1: for r,p in (SELECT ratings.bookID, ratings.binaryrating from ratings,feature_documents WHERE ratings.bookID=feature_documents.bookID AND ratings.userID=$DBbook_userID) loop
       –CONSTRUCT THE USER FEATURE VECTOR
       if p=1 then
       uvec:= (SELECT MADlib.svec_plus(uvec, (SELECT vec_rep FROM feature_documents WHERE bookID = r)));
       else
       uvec:= (SELECT MADlib.svec_minus(uvec, (SELECT vec_rep FROM feature_documents WHERE bookID = r)));
       end if;
       end loop;
2: Task2: for r in SELECT bookID, 1- MADlib.tanimoto_distance(vec_rep, uvec)
       FROM feature_documents
       WHERE bookID=$DBbook_itemID loop
       return next r;
       end loop;
3: Task3: for r in SELECT bookID, 1- MADlib.tanimoto_distance(vec_rep, uvec)
       FROM feature_documents
       ORDER BY 2 DESC LIMIT 50 loop
       return next r;
       end loop;
```

listing 1. We tried a variety of similarity/distance metrics (cosine, euclidean distance, etc), but the best results were obtained by calculating the Tanimoto distance which is is based on the ratio between the size of the intersection of two vectors by the size of the union. Table 1(a) shows the summary of the submission results as reported by the evaluation service.

## 4 Task 3 experiments

Task 3 dealt with the diversity of the items produced in a Top-20 recommendation list for each user with respect to the author and subject properties of the items. The recommendations lists were computed by considering all unrated items by each user and selecting the Top-20. Even though the required format of the submission was of the form "$DBbook_userID/$DBbook_itemID/score", the ranking of the items by their score within the Top-20 list was not taken into account by the evaluation system. Instead the evaluation metric considered the participants' positions in the respective F-measure@20 and ILD@20 (as defined in footnote[3]) rankings. More specifically, first a ranking with all participants was generated according to their ILD@20 values. A second ranking with all participants was then generated according to their F-measure@20 values. For each participant the score was finally computed as the mean of the positions in the first two rankings.

Listing 3 in Algorithm 1 shows how we calculated within the database the Top-N recommendations for unseen items for each distinct "$DBbook_userID" in the evaluation set. Again the semantic feature vector for each user is calculated as in listing 1. As in Task 2, we tried a variety of similarity/distance metrics and the best results were still obtained by utilizing the Tanimoto distance. In order to account for the diversity of the items in the list we initially produced a larger Top-50 list of items and we then measured all the pairwise similarities between those 50 items in the 10125-dimensional feature space. The items that appeared more frequently to exhibit zero similarities

| **Task 2** | Submitted |
|---|---|
| P@5 | 0.6225 |
| R@5 | 0.4662 |
| F1@5 | 0.5331 |
| | |

(a)

| **Task 3** | Submitted | Corrected |
|---|---|---|
| P@20 | 0.0228 | 0.0134 |
| R@20 | 0.0738 | 0.0447 |
| F1@20 | 0.0348 | 0.0206 |
| ILD@20 | 0.4447 | 0.4635 |

(b)

**Table 1.** Results on the evaluation set for Task 2 (a) and Task 3 (b)

with their peers finally made it to the Top-20 list. That list was then ordered by the similarity of the items to the user's feature vector (even though that was unnecessary as explained above). Unfortunately due to a bug in the driver script, all the initial items in the Top-50 list were take into account during sorting (not just the Top-20 more diverse items), which resulted in just keeping the first 20 items from the initial Top-50 list, and therefore neglecting the diversity calculations. The bug was discovered just a few hours after the Challenge's deadline, but since the evaluation service was still accepting solutions we report it here for completeness[4]. It is interesting to note the increase in the ILD@20 value of the corrected submission which comes of course at the price of a drop in the F1@20 score. Table 1(b) shows the summary of the submission results as reported by the evaluation service both for the last result submitted within the deadline, and the bug-corrrected result.

## 5 Conclusions

In this paper, we invesigated the exclusive use of semantic technologies for Tasks 2 and 3 of the ESWC'14 Challenge on LOD-enabled Recommender Systems, combined with an efficient in-database analytics approach for efficiently producing the recommendations. Our results indicate that the utilization of semantic technologies can perform well on a variety of recommendation metrics. This makes interesting the prospect of further investigating hybrid algorithms between collaborative filtering and semantic approaches, that would effectively complement each other, and thereby improve the recommendation performance.

## References

1. Koren, Y.: The BellKor Solution to the Netflix Grand Prize. Available from http://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf (August 2009)
2. Toscher, A., Jahrer, M., Bell, R.: The Big Chaos Solution to the Netflix Grand Prize. Available from http://www.netflixprize.com/assets/GrandPrize2009_BPC_BigChaos.pdf (August 2009)
3. Pazzani, M., Billsus, D.: Content-based recommendation systems. In Brusilovsky, P., Kobsa, A., Nejdl, W., eds.: The Adaptive Web. Volume 4321 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2007) 325–341

[4] Final reported ranking for the teams is based on results submitted up to the deadline