Studying the influence of implicit feedback on TV recommendations with Linked Data patterns

Stefan Perez
Artificial Intelligence
2504661 / spz600
stefan.perez@gmail.com

ABSTRACT

The amount of data available on the web is growing each day. Each day new TV programs are created and new information comes available about these programs. Choosing what to watch becomes more difficult due to the large amount of TV programs that are available. Therefore, the need of a recommender system is preferable to cope with the large amount of data. Traditional recommender systems are mainly focussing on recommending items to users that are closely related to their preferences. A limitation of those approaches is that they recommend items which will never break the filter bubble. Users only get recommendations that are closely related to the user profile. Another problem that traditional recommenders are facing is the sparsity problem. Those recommenders are relying heavily on their (meta)data and often this (meta)data is incomplete or missing. The need of a better approach and solving this relevance and sparsity problem is needed. Therefore a Linked Data (LD) recommender is created which combines traditional recommender (e.g. content-based and collaborative filtering) approaches with LD. The LD recommender uses LD to semantically enrich the TV programs and based on the generated patterns new relations between programs were made.

Different algorithms are been tested to define which algorithm outperforms the other. Each algorithm has implicit user log feedback available. During the experiments we found that using implicit negative feedback could indeed increase the relevance of the recommendations regarding the testset. We found that within two given scenarios the false positive rate, the rate that shows how unexpected (not liked/viewed) programs influence the recommendations, is quite high. Which indicates that implicit negative feedback indeed have a high influence on the recommendations.

Additional, a more generic analysis, we found that for a quite large group (24,68% of the total users) we could create relevant recommendations. Despite this, using LD also introduced unexpected results (e.g. a BBC One program got connected to a CBeebies program) which might be useful to increase serendipity. Nevertheless, the outcome of this research gave enough insight on how important implicit (negative) feedback can be used in order to provide relevant recommendations with LD.

General Terms

Recommender systems, Linked Data, Ranking, Patterns

Keywords

RDF, Social Web, LD, Patterns, Ranking algorithm, Recommendations, Hybrid recommenders, TV, BBC

1. INTRODUCTION

Searching for a program, movie, music artist or any other media item is becoming more common than before due to the uprising technology regarding recommendation systems. Companies as Netflix[1], Amazon[2], Spotify[3], Google[4] and many more are focusing on providing the best possible results to a user regarding their preferences. Terms as: "because you like this, you may also like this" and "other users that bought this also bought this.." are nowadays notions that famous companies use to recommend items to their users. These companies are seeking to predict the rating or preferences that the user would give to a specific item.

Every day we are using media to entertain ourselves, to get informed or to socialize with others. The media is becoming an important factor within our lives. Almost everybody have a mobile phone or a tablet which is connected to the internet. With the uprising technology features as watching online television (TV) is becoming more popular. The BBC created for example, the BBC iPlayer[5]. Users are now capable of watching TV online and as we investigate the statistical use[6] of the BBC iPlayer we see that last August of 2015 226 million request were been made to watch TV content on the BBC iPlayer. Which is on a daily base an average of 9 million requests. To increase, for example the number of requests to the BBC iPlayer or to improve the experience users have with an online media tool, relevant recommendations are needed to guide users in coping with the large amount of choices a user is facing.

Generating a recommendation list can be done in several ways, the most popular ones are the collaborative filtering method and the content-based method. The difference between those two methods are in the way they treat data. A collaborative filtering recommender mainly focus on recommending items to users based on the similarities of behavior between users. While content-based recommenders are recommending items to users based on similarities between con-

[1]`http://www.netflix.com`
[2]`http://www.amazon.com`
[3]`http://www.spotify.com`
[4]`http://www.google.com`
[5]`www.bbc.co.uk/iplayer`
[6]`http://downloads.bbc.co.uk/mediacentre/iplayer/iplayer-performance-aug15.pdf`

tent items in user preferences and user interaction[18, 17]. For example, Netflix is a company that use content-based filtering method to feed their recommendations. They try to suggest possible recommendation that fits the user profile. If, for example, the user likes a specific movie having the genre: "comedy" than Netflix will suggest other comedy related movies. The technique behind this is that they match properties based on the content of that movie and the interaction of the user[1]. Nowadays we see a shift of techniques used in recommendation systems. Neither one or the other alone works best for generating recommendations. Companies as Netflix and Amazon are not mainly focusing anymore on only one technique. They try to take the advantages of both techniques and are trying to implement those. Whenever a recommendation system use both techniques, to generate recommendations, it is labeled as a hybrid recommender[5].

With regard to the current recommendation systems they have one major problem: "sparsity[8, 5]". Often the (meta) data that these current recommenders are using is incomplete or missing. Which might cause the effect that only the very popular and best described items are recommended. Where perhaps other not very popular and well documented items might also be interesting to recommend. For that reason, a new promising approach could solve that problem: a Linked Data recommender(LD)[7].

With a LD recommender the problem of "sparsity could be solved by interlinking to other available datasets. The data could be semantically enriched and new relations and connections become available. Nevertheless, using LD also introduce a new problem: relevance. Interlinking towards other datasets adds new information to the data which might introduce noise. Therefore to increase the relevance a filtering method is needed to define which programs are relevant and which are less relevant.

In this research a LD recommender is created that tries to increase the relevance by applying a pattern calculator and a ranking method. Three different algorithms are compared and are statistically been tested based on the precision-recall ratio. We will test several scenarios (negative feedback, positive feedback, negative/positive feedback) to define the influence of implicit negative feedback on TV recommendations. The influence depends on the precision value regarding the recommendations. Each algorithm has as input semantically enriched BBC TV program data. From this data, patterns are created which are used as input for the algorithms. The goal of each of those algorithms is to increase the relevance and trying to burst out of the "filter bubble[15]" and provide recommendations that breaks the invisible barrier of items that are not presented to the users. In the end, a conclusion could be made whether, with use of LD, implicit negative feedback influence the recommendations regarding TV programs.

In short, in this paper a research is been presented that explains an approach to TV Recommendations that combines the use of Linked Data Patterns with a hybrid recommendation approach. We generate content-based recommendations and combine those with a collaborative approach using popularity statistics and user ratings. With different algorithms we test the influence of patterns on the recommendations and are solving the problems that current recommenders are facing: sparsity and relevance. The paper will continue with the related work section, then section 3, which will explain which data is been used. Section 4 will explain the methodology that is used and section 5 will explain the experimental setup. Section 6 will evaluate the results of the recommendations. Section 7 will discuss the limitations and future work. The paper ends with the conclusion, section 8.

[7]`http://www.linkeddata.org/`

2. RELATED WORK

The first known recommender system date back to the 90's where information retrieval was the upcoming stream[7]. The amount of data back then was growing each day and nowadays it is still growing. To process all these data is a difficult job and therefore the need of filtering methods was desirable[21]. Filtering the data helps to provide results that are the most valuable. Suggesting items to a user is a decision making process where specific items are suggested regarding the preferences of the user[18]. Nowadays, the goal to present the most valuable result to the user can vary from increasing the user satisfaction till increasing the number of items sold. To suggest such results different techniques can be used. The most famous techniques are collaborative filtering and content-based filtering[21, 18, 2].

2.1 Collaborative filtering

With the collaborative filtering method items will be recommended based on what other user liked in the past. They based their recommendations on the similar taste of that active user[21, 18, 2]. Amazon is a good example of a company that use collaborative filtering to recommend items to users. The algorithm that they use, to recommend items, checks customers that are the most related to the active user and find items that other users liked to recommend new items[10].

Digging deeper into the algorithm that Amazon nowadays use, an important note can be made. They used the basics of every standard collaborative filtering method but they extended by adding the item-to-item functionality. The item-to-item collaborative filtering functionality try to match each of the users purchased and rated items to similar items. After this process is been completed they try to combine those similar items into a recommendation list and output it to a user[11]. The most similar items will be suggested based on finding items that a specific user bought together. If, for example, a user in the past bought two books: "an introduction to recommendation systems" and "an introduction to information retrieval" then whenever a new user search for the book: "an introduction to recommendation systems" the item-to-item collaborative filtering functionality will recommend to also buy: "an introduction to information retrieval". Storing all these information and filtering out the preferences of the user is computationally expensive. Especially when there is a user that rates a large set of products and buys a large amount of these products. Calculating these item-to-item matches will be expensive because it will grow exponentially[10].

2.2 Content-based filtering

Content-based filtering method is a method that takes the properties of the item as their guideline to recommending items[21, 18, 2]. Which means that whenever an item contains a set of properties (e.q. genre, duration etc) it tries to find other items that are closely related to that item. Netflix is a company that applies this content-based filtering method to recommend new items. In a techblog[1] published by Netflix they explained how their content-based filtering algorithm operates. First, when Netflix was a starting company they organized a coding competition where they asked to help improving their rating and ranking predictions. They rewarded a price for the person who achieved the highest possible rank. They encountered two problems generating good movie rating predictions and ranking those predictions. For generating good movie predictions they putted their focus on improving the similarities between programs. They look if they could find any similarities between programs but also between users. Which indicates that they also add some collaborative filtering functionality to their system. The expected result was that if a user likes a specific movie which is labeled as genre "comedy" then then the system try to predict other movies regarding the genre "comedy".

The other problem was ranking the predictions. Presenting the most important results in a ranked list is in most common recommenders implemented. Netflix wanted to present the best matching result compared to the user and movie profile. An important measurement they used is the popularity score. The more users rated a specific movie the more likely this movie will end up high in the result list, depending on its rating score.

To determine which approach is the best is not important anymore. Nowadays, we see that neither one or the other alone works best. Popular companies as Netflix and Amazon are implementing features from both approaches to improve their recommendations and to increase their users satisfaction[5].

2.3 LD recommenders

The techniques regarding recommender systems are growing by each day. New techniques are explored and the web is become a place where all the data is becoming more reusable. Due to the introduction of the Semantic Web and Resource Description Framework (RDF), structured data became available and freely accessible to everyone. The advantage of publishing the data online made it possible to interlink between data sources. The data will actually get enriched by other data sources[4].

The introduction of the Semantic Web made it possible to interlink data sources. The advantage of interlinking data sources is that the information stream will automatically rise. That means that the data does not have to be written by the program but can be used by linking the own dataset to those other data sources. The effect of LD on recommenders are been researched in the past few years and a rise of LD recommenders is becoming more on the web.

In most LD recommenders, that are available online, there is not a clear strategy available which can be labeled as content-based filtering recommender or collaborative filtering recommender. In most cases the LD recommenders will use both strategies to recommend items, those are also called Hybrid recommenders[3, 5].

One good example where LD is been used to generate recommendations is by Maccatrozzo et al[14]. They showed that by using LD patterns the outcome of the recommendation could be very accurate towards the test data that they had available. They used as dataset DBPedia[8] books and got user ratings for a large amount of books which were represented within the testset. For each book that the user liked they generated patterns based on user ratings and recommend other books based on the user ratings. What they showed is that to which extend the patterns are generated to which item property. That means that it is possible that for a specific property adding a semantic pattern is not effective and therefore the choice of which external vocabularies to choose is also important to note.

Another recommendation system that uses LD which is closely related to the approach that we want to implement, is dbrec[16] created by Passant. This system generates music recommendations based on LD. They created recommendations for 39.000 bands and solo artist. To generate those recommendations they calculated the semantic distance between two resources. The semantic distance is calculated through indirect and direct links between two resources. Besides calculating the distance they also generated a recommender algorithm that takes this distance as one of their input parameters. The results of the recommendations were quite interesting. As they mentioned in their paper, the outcome of their novel recommendations was quite high. They discovered that many of the recommendations were unknown to the users: 62% of the test group did not expected the results.

2.4 Implicit feedback

As the title of the paper stated the recommendations are been tested against implicit user feedback. The feedback is stored within a large database and this data contains information of a full three month timespan[9]. The hard part of having implicit feedback is that the link between user feedback and user preferences becomes less clear. Information about whether the user liked a specific TV program is hard to say with implicit feedback. Therefore, several methods are been tested to define which setting outperforms the others (e.g. which settings fits the user preferences best with respect to the implicit data).

There has been a large amount of research conducted towards implicit feedback. Mostly those researches only focusing on positive implicit feedback. In an overview study conducted by Keely and Teeven[9] only 4 of the 27 techniques that were used focused on implicit negative feedback. The aim of this research is to solve the problems that current recommenders are facing by solving the sparsity problem with LD and increase relevance by using implicit (negative) feedback together with LD patterns. Doing several experiments will show whether implicit negative feedback in combination with LD indeed increase the relevance of the recommendations regarding BBC TV programs.

The approach that we will implement is fundamentally equal to these LD approaches. However, we do not calculate item to item similarity but we will use the Linked Data graph structure to generate recommendations. We are seeking to connect items through LD patterns which is a different approach than other LD recommenders do. Our approach tries to find potentially new links between items. In addition, in our approach we analyze LD sources by using knowledge patterns. These knowledge patterns are a

[8]http://wiki.dbpedia.org/

[9]A more detailed explanation of the data can be found within section: Data

tool which represent semantic heterogeneity of the ontologies that will be used. With all these extensions and state of the art approaches that we implemented, we aimed to find new connections between TV programmes through semantic patterns. As an extension we not only generate semantic patterns we also compared different algorithms to define which algorithm could improve the recommendation quality (e.g. increase relevance).

3. DATA

The data that is been used for this project comes from the BBC as we are focussing on generating recommendations for the TV industry. The BBC have all their structured data available online and can be retrieved in several formats. Which comes in with a few advantages, one advantage is that due to the fact that they provide program data in RDF the enrichment process of the programmes can easily be done by mapping it to other Linked Data sources. In the following sections an overview will be given about the structure of the TV programs data.

3.1 Structure of the data

A program of the BBC can be viewed in several ways, where each program can be retrieved by their unique identifier. The data is available online and can be searched by the following URL: `http://www.bbc.co.uk/programmes`. It is possible to add an extra parameter to the url in case a different format (e.g. .json,.xml or .rdf) is required for a specific program. Each program on the BBC website is defined by their unique identifier, called: ”pid”. To view a program on the BBC website this extra parameter is needed to retrieve the desired program. In short, to view information — about the program ”top gear”, for example — within a xml format the following url should be defined: `www.bbc.co.uk/programmes/b006mj59.xml`. To retrieve a program in specific format, different formats can be entered within the URL.The formats that are available are: .xml, .json and .rdf. Each program can be outputted in one of those three types. In general, viewing information about a program on the BBC website within a specific format the need of adding the pid and format variable to the URL is required.

There are few prerequisites that are needed to generate LD recommendations. One prerequisites is that we needed the data within a semantic format: RDF. The BBC has all their TV programs available within this format. Therefore, these programs were suitable as input for our LD recommendations.

The BBC also stored user interaction with the iPlayer[10] during a specific timespan. Within this logs a overview can be found of what program a user watched and for how long they watched. Each time that the user watched a different program a new session is been recorded in the log.

To generate LD recommendations we needed all the data from the programs in RDF format. To test the recommendations a specific timespan was chosen to determine which programs should be used as input for the the recommendations. The programs that were used for recommendation purposes were aired between the 1th of march 2014 till the 31th of may 2014. The three months timespan made it possible to accurately test the outcome of the recommender

[10]`ww.bbc.co.uk/iplayer`

and check if the results were expected by comparing it with the user logs. Within the user logs we had data available of those three months. Within this dataset the amount of viewing time per session was recorded per user. For each program that the user watched a new session was created. of the amount of viewing time per program within these three months.

Looking deeper into the data of a programme,aired by the BBC, the following properties can be defined:

- PID (unique identifier id)
- Title (the title of the program)
- Synopsis (short, medium and long synopsis)
- Ownership (e.q. BBC one, BBC two etc.)
- Genre
- Format
- Broadcast date
- Duration of the program (only if available)

Each program has a unique identifier which enables to search for a specific program based on their identifier. A program can contain several formats and genres. For example, the program: ”The apprentice[11]” is a program aired by the BBC One and contains two genres: ”Factual” and ”Entertainment” and two formats: ”Reality” and ”Talent Shows”. A total overview of the structured data of this program can be found in the Appendix.

3.2 Generating patterns

To generate patterns the data should be in RDF format. As explained in the previous section we gathered the BBC data in that desired format. Within the user logs of the BBC we selected a specific timespan: 01-03-2014 / 01-05-2014 (3 months), and from this timespan we started generating patterns. We gathered from those three months 1525[12] unique programs. We used these programs, as described in detail in the methodology section, as input for our semantic enrichment process and for generating the patterns.

To generate patterns three steps were conducted, as can be read in more details in the paper of Maccatrozzo et al[12], in order to generate semantic patterns. The first step focused on the enrichment of the synopsis. The concepts of each synopsis was extracted with the help of the NERD API service[20]. This service makes it possible to annotate concepts based on twelve named entity extractors.

In the next step the same process of extracting concepts was executed. But instead of extracting concepts out of synopsis, concepts were extracted from the title and credits. The enrichment of the title and credits was performed by looking for any URIs which describe the programme or the people (e.g. directors, actors) involved in it. Also this process was executed by the NERD API service.

Whenever the previous two steps were executed the final and last step could be applied: pattern extraction. A pattern, which will be explained in more detail within the Methodology section, is basically a generalized path within

[11]`www.bbc.co.uk/programmes/b0071b63`

[12]see appendix B for the SPARQL query retrieving unique programs

a graph which is described by the number of links and type of nodes. A pattern was created based on the number of links and type of nodes that two programs share with each other.

3.3 BBC Logs

To test the recommendations we gathered, from the BBC, user logs which contains information about the total amount of viewing time per session, per program, per user. For every program that the user watched or clicked the amount of viewing time is recorded together with belonging PID. This value may arise from zero (where it can be assumed that the user, for example, clicked the program and closed it) to the maximum available time that is stated for that specific program.

The data of the logs are used as input for the experiments and as output to verify the results of the algorithms. For each user we divided the log entries into a 60:40 ratio. Which means that 60% of all the data that is available, within a specific timespan, is used as a training set (=of the To test the experiments a division is been made, per user, to define the training and the testset. For each user we selected a ratio of 60:40. Which means that if a user x watched ten programs than six of those programs are stored within the training set and four are stored within the testset. For each user we could use the programs, that were stored within the training set, as input for finding our pattern process.

For example, the training set and the testset of user X could contain the following items:

Table 1: Example data training set

Trainingset: User X		
SessionID	PID	Viewing time (in minutes)
01	b0139jv6	84
02	b03xk3r9	34
03	b03y463g	8
04	b03yfz3k	12
05	b03yfz86	1
06	b03yx0sl	17

Table 2: Example data test set

Testset: User X		
SessionID	PID	Viewing time (in minutes)
07	b03yx0sl	9
08	b03yvn8r	2
09	b03ymfs4	16
10	b03ywxjs	0

Those tables are been generated for each user individually and are the input for the recommendation process which is explained within next section.

4. METHODOLOGY

The Vista-TV project is collaborative european-union funded project where the VU University is part of it. The main focus of the project is to create real-time TV recommendations. They are analyzing anonymized viewer behaviour and try to create real-time recommendations. The project NoTube is a similar project where the VU participates in. This project tries to demonstrate how to use Semantic Web technologies to connect TV content and the web through Linked Open Data. To enrich the TV programmes metadata we used a semantic approach, developed in the context of the Vista-TV project[13] [12, 22].

To elaborate, the enrichment approach, that is used within this research, consist out of three components based on the approach by Maccatrozzo et al[12]: programme metadata reader, the enrichment of tv programs and a ranking algorithm. Figure 1 gives a total overview of those components:

4.1 Step 1: Programme metadata reader

The first step is gathering all the programs from a specific timespan. The BBC provided us a database file with all the programs that were aired during a specific timespan. Within this database all the programs were stored plus all the user activities. We created a tool that gathered all the metadata of every TV program from the BBC website. For every program a call is been made to the BBC website and for every program the metadata in RDF is stored within a RDF Store. We only stored the important properties of every program: titles, credits and synopsis. Subsequently, all the BBC programs are used for the next step where the programs will get enriched semantically.

4.2 Step 2: Enrich TV programs data by mapping it to Linked Data sources

The second step is to enrich the retrieved data with LD sources and extract patterns. We used several general and domain vocabularies to find paths between items of programs[13]. The domain and general vocabularies are part of the annotator tool that is used called NERD[14]. Those vocabularies operating via annotators, which means that they annotate specific concepts within the titles, synopses and credits. Annotators are tools that annotates certain sections in documents/text as referring it to different entities or to categories. Each annotator can have their own domain and specialist knowledge. Geo annotators mainly focussing on annotating geo related concepts and referring it to geo related categories whereas cultural annotators might focus more on the historical perspective. Due to the fact that TV programmes can cover a wide range of topics and categories, searching for a specific annotator is very difficult. Therefore we used in this research the NERD[15] tool which uses 12 annotation tools and aggregates the results.

Whenever we had a title, for example: ”Reggie yate's extreme South Africa[16]” which was aired on the BBC Three channel, the annotators annotated the following concepts out of this title: ”Reggie yate's”, ”extreme” and ”South Africa”. The big advantage that we had, by using those annotators and mapping it with other LD sources it, was that we added more information to the source than we had beforehand. When we used this program as input for our semantic enrichment process we found a pattern with the program: ”The

[13]http://vista-tv.eu/

[14]The different domain and general vocabularies are explained here: http://nerd.eurecom.fr/documentation#extractors

[15]http://nerd.eurecom.fr/

[16]http://bbc.co.uk/programmes/b03x1zs2

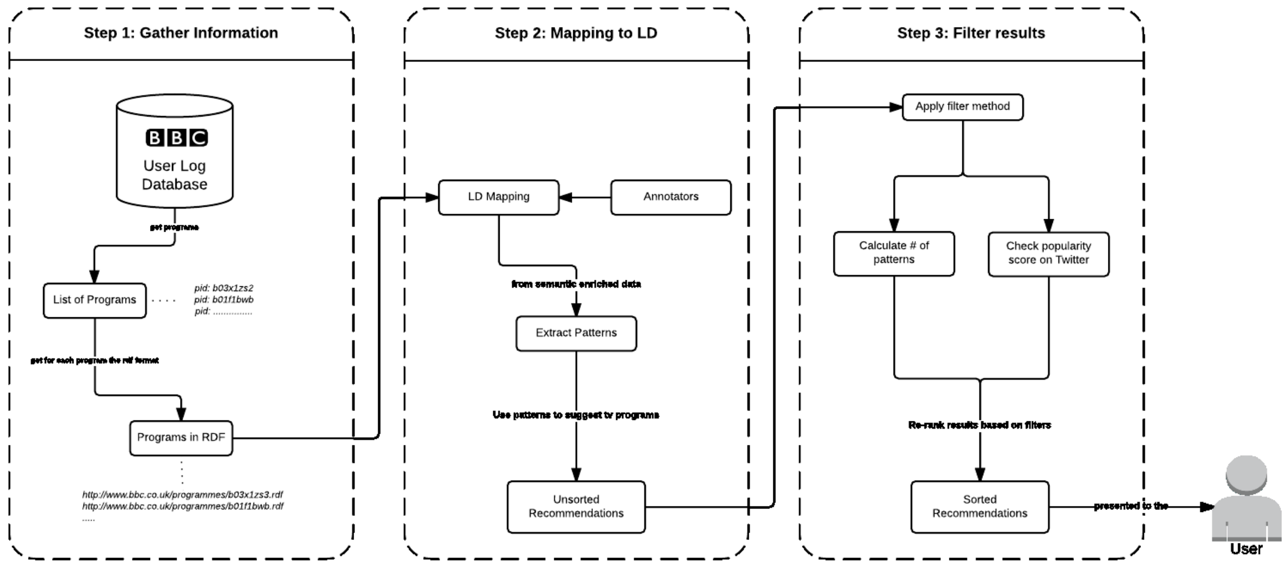


Figure 1: Workflow of our BBC TV Recommender

Sky at Night[17]" which is a science and nature program aired on the BBC FOUR. Comparing the metadata from both programs does not show a clear relation between the programs. Consequently, traditional recommenders will not recommend those programs. The reason why there is a pattern between those two programs lies within the title of the program. The annotators annotated the word "extreme" as an concept and based on that concept patterns could be made, which is visible in figure 2.

The programs are connected through a pattern because the annotator annotated the word: "extreme". Mapping this concept to DBpedia the music band: "extreme" was found. The band "Extreme" is influenced by another music band called: "Queen" and a member of this band is "Brian May". Finally, we found that "Brian May" was a guest at "The Sky at night" show and therefore a connection could be established between the programs "Reggie Yate's extreme South Africa" and "The Sky at Night".

The patterns are generated based on the DBpedia concepts which are defined in RDF. All the data that is available within DBpedia is based on the content of the pages within Wikipedia[18]. The created patterns between the two programs, within the given previous example, can also be visualized as can be seen in figure 3.

In figure 3 the same example pattern is used but described based on content pages of Wikipedia. The concepts are annotated in yellow and are as goal to clarify why the pattern is created. The annotated concepts within the Wikipedia pages are the reason why there is a connection created between the two programs. The data on the Wikipedia pages is written in plain text but the data is also available in a structured format which made it possible to process the data semantically. The structured data will be investigated and been annotated by the semantic annotators that we used.

[17]`http://bbc.co.uk/programmes/b03y31f0`
[18]`https://en.wikipedia.org/wiki/Main_Page`

For a more detailed explanation of the enrichment of TV programs we refer the reader to the work of Maccatrozzo et al [12].

4.3 Step 3: Ranking

In this last step we need to filter the patterns and apply a function which might improve the relevance of the recommendations. If we generate recommendations only based on the generated patterns we randomly provide recommendations to users which are not ranked. Therefore, we need a specific sort mechanism to calculate the ranking. We applied the fundamentals of the Weighted Rating (WR) algorithm created by IMDB[19] to rank the recommendations. The input values for the Weighted Rating algorithm are generated based on two sub components:

1. Number of patterns: A function that calculates the number of patterns that connects two programs with each other.

2. Twitter Popularity checker: A function that checks the popularity of the recommended programs on Twitter based on the number of tweets there are published regarding the pids (e.g. b03y31f0 etc).

We will now explain in detail how these two sub components operate:

1. Number of patterns. After the process of generating the patterns is executed we could make recommendations to users. To filter out interesting results for a specific user a filtering method is needed. One method we used is calculating the number of patterns there are between two programs. The number of patterns represents the number of connections there are between two items, in our case two TV programs. If we take the same example given in step

[19]`http://goo.gl/GrGUqH`

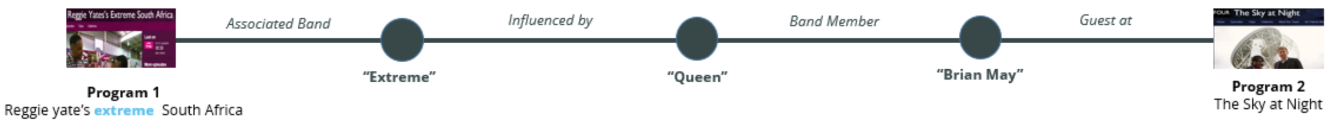
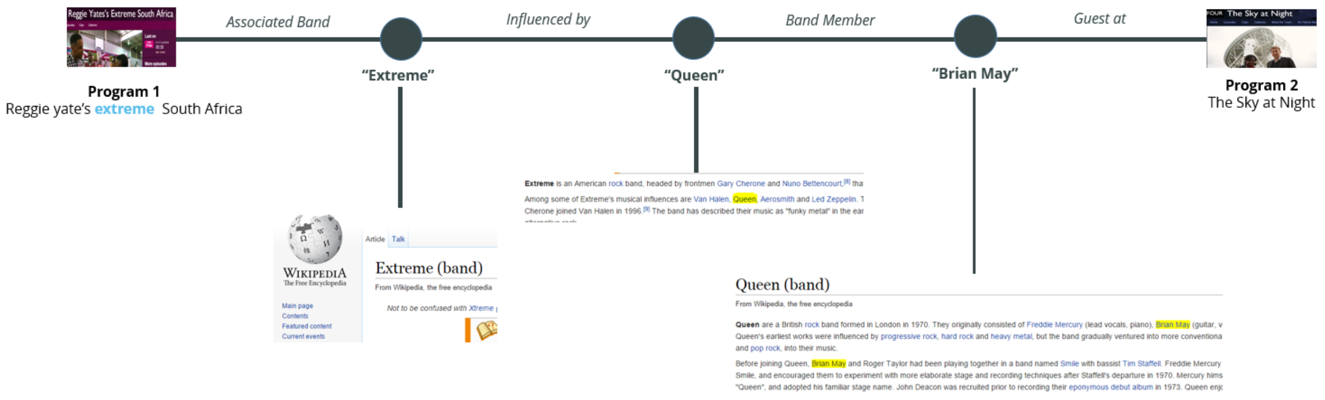


Figure 2: An example pattern between two BBC programs



Extreme (band)

From Wikipedia, the free encyclopedia

Not to be confused with Xtreme

Extreme is an American rock band, headed by frontmen Gary Cherone and Nuno Bettencourt,[8] tha

Among some of Extreme's musical influences are Van Halen, Queen, Aerosmith and Led Zeppelin. T
Cherone joined Van Halen in 1996.[9] The band has described their music as "funky metal" in the ear

Queen (band)

From Wikipedia, the free encyclopedia

Queen are a British rock band formed in London in 1970. They originally consisted of Freddie Mercury (lead vocals, piano), Brian May (guitar, v
Queen's earliest works were influenced by progressive rock, hard rock and heavy metal, but the band gradually ventured into more conventiona
and pop rock, into their music.

Before joining Queen, Brian May and Roger Taylor had been playing together in a band named Smile with bassist Tim Staffell. Freddie Mercury
Smile, and encouraged them to experiment with more elaborate stage and recording techniques after Staffell's departure in 1970. Mercury hims
"Queen", and adopted his familiar stage name. John Deacon was recruited prior to recording their eponymous debut album in 1973. Queen enj

Figure 3: An example pattern between two BBC programs explained via Wikipedia content

two of this section, we could calculate the following number of patterns, which is is described within table 3.

Patterns with programs:	
PID	Title
b03y31f0	The Sky at Night
b01f1bwb	Southern Rock at the BBC

Table 3: Result of programs that were connected with: "Reggie Yate's extreme South Africa" through patterns

Within this example only two BBC programs were connected through patterns. Logically we will recommend both items to watch but even with this example we want to apply the pattern calculation method. The pattern calculation method checks the amount of patterns between two programs. The higher the amount of patterns between two programs the more connected they are through semantic enrichment. The number of pattern is stored and will be used for the second component of our ranking algorithm. In table 4 an overview is given of the calculated patterns.

Number of patterns	
Title	# of patterns
The Sky at Night	2
Southern Rock at the BBC	7

Table 4: Calculated patterns based on the program: Reggie Yate's extreme South Africa

Based on the number of patterns a ranked list is created where in above example the first rank is taken by "Southern Rock at the BBC" and the second rank by "The Sky at Night". The output of this component will be used as input for our second component:

2. *Twitter Popularity checker.* We wanted to use the collaborative method to check the popularity score on a social media platform. We choose to use Twitter as our popularity measure tool because other social media platforms as Facebook or Instagram does not provide enough information for every BBC TV program. The number of tweets about a specific BBC TV program was the measure of how popular a program was. For each program within the recommendation list we made a call to the Twitter API[20] and retrieve the number of tweets about each program within recommendation list. In the end we keep track of a list for every program within the recommendation list with the number of patterns and the Twitter popularity score. We apply at the end a specific algorithm to calculate a weighted score and rerank the list.

The results of both components were used as input values for the last step of our ranking process: ranking the results based on the weighted rating function. The fundamentals of the function were defined by IMDB. We modified the function and achieved to have it working with our patterns and social media scores. We defined the function as follows:

$$WR = \frac{pR + mC}{p + m} \tag{1}$$

Where:

- R = Popularity score of the TV program on Twitter = (Rating)
- p = number of patterns between two programs = (Patterns)
- m = minimum patterns needed to be listed in the Top 10

[20]`https://dev.twitter.com/`

- C = the mean popularity score across all the TV programs

The minimum number of patterns needed is been set to five. This number represent the average of all the patterns that are generated between programs compared to what users viewed. This value will not change unless the average p value for a specific user is lower than five, then it is set to one. The mean popularity score is variable and will be calculated per user, depending on the number of viewed programs. For the use case of the "Reggie Yate's extreme South Africa" TV program the following WR is calculated:

Weighted Rating (WR) scenario		
PID	b03y1f0	b01f1bwb
R	36	12
p	7	12
m	5	5
C	24	24
WR	31	15.52

Table 5: The calculated Weighted Rating (WR) for the "Reggie Yate's" scenario

Based on the WR score the following ranked list will be generated:

1. b03y1f0 (WR = 31)
2. b01f1bwb (WR = 15.52)

The calculated WR score is the guideline of sorting the recommendations. The result that has the highest WR value will be presented to the user as the first recommended result. The WR value will always float between 0 and 100. The minimum amount of patterns needed to be listed in the Top 10 is set to five. This is a fixed value for all the programs unless the average number of patterns is below the value five. In that case the value is set to one.

In appendix C an overview of these three components is given in pseudocode.

5. EXPERIMENTAL SETUP

5.1 Goal

The main goal of this paper and of this experiment is to check whether implicit negative feedback influence the recommendations regarding the testset. With this main goal we might prove that implicit negative feedback can indeed improve the recommendations. Based on the semantic enriched TV programs and the generated patterns, recommendations could be created that are been tested against user logs gathered from the BBC. The pre-collected dataset of user logs made it possible to distinguish per user what he or she watched and for how long they watched it. It made it possible to select a specific user and checks what he or she watched within a certain timespan. For each user the timespan was cut in half resulting in two parts: a training set and a testset. The recommendations were generated for the training set and are tested against the testset. The result of these test are gathered and described in section: "Results". With this experiment the following research question will be answered: "What is the influence of implicit negative feedback on LD TV recommendations".

5.2 Data overview

In figure 4 a percentual overview is given which represent the number of patterns that were generated based on the BBC data.

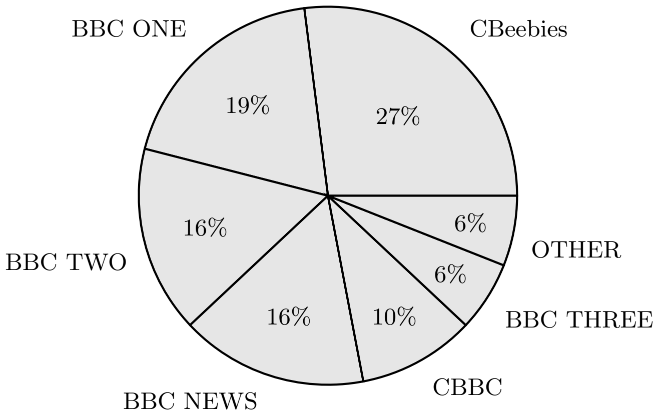


Figure 4: Percentual demographic of the generated patterns per channel

The values within the demographic represent a percentual image of the created patterns. The total amount of programs used were 1525 programs (100%).

Investigating the outcome of the numbers of patterns the following remarks could be made:

- 37% of the generated patterns were CBeebies and CBBC programs (564 programs)
- The total amount of pattern programs that were collected was 1525 where the amount of programs within the database was 6588

Based on these findings we checked within the user logs the viewing history of users that watched a CBeebies program or a CBBC program. We found out that in most of the case (96%) users that watched programs on the CBBC and/or CBeebies channel did not change to other channels. In result to that, for the experiment the patterns for the CBeebies programs and CBBC programs were not used. In addition, we excluded all the users within the user logs that watched any CBBC & CBeebies program. It has one main reason why those patterns were excluded:

- If we used those patterns then we will not burst out of this filter bubble. Which means that whenever a user watched a CBEEBIES program there is a very high opportunity that the recommender recommends a similair CBEEBIES or CBBC program.

To conclude, we focussed mainly on the programs that were aired by all other channels except for those of CBeebies and CBBC.

Sample size	
Confidence level	95%
Confidence interval	5
Population	6434924

Table 6: Sample size of the whole population

	Recommended	Not recommended
Preferred	True-Positive (tp)	False-Negative (fn)
Not preferred	False-Positive (fp)	True-Negative (tn)

Table 7: Classification of a recommendation

6. RESULTS

The main goal of the experiment was to check what the influence of implicit feedback is on patterns TV recommendations. The recommendations were compared against the testset of a user profile. The trainingset was used as input for our recommendations. The following scenarios were tested:

- liked (viewed) programs
- weighted liked programs

Within the first scenario only the programs that the user actually watched, for longer than zero minutes, were used as input for the recommendations. If a user watched three programs then we gathered for these three programs the patterns. In addition we ranked the results based on the created ranking algorithm.

In the second scenario a selection mechanism is been introduced to classify whether a user liked the program or not. For every program, the viewing time is compared to actual duration of the program.

To compare which method worked best in the case of comparing different scenarios by checking if negative implicit feedback influence TV recommendations, we applied the precision-recall functionality.

6.0.1 Data Input

To analyse the outcome of the created patterns we took a sample size set of the total amount of available user logs. The sample size is been set to 1067 and is calculated based on the following parameters which can be seen in table 6.

Within this sample size set a selection of users were made which watched programs that were connected through other programs by patterns. If a user watched several programs but not one of those programs did not contained any patterns then this user was excluded from the test.

Each scenario is been tested against the precision-recall ratio defined by Gunawardana et al [6]. Each scenario is classified based on this ratio. The ratio is calculated based on the following settings, which can be seen within table 7. The precision and recall ratio is calculated as follows, which is also defined by Gunawardana:

$$precision = \frac{\#tp}{\#tp + \#fp} \tag{2}$$

$$Recall(TruePositiveRate) = \frac{\#tp}{\#tp + \#fn} \tag{3}$$

$$FalsePositiveRate(1 - Specificity) = \frac{\#fp}{\#fp + \#tn} \tag{4}$$

In the following sections different scenarios are been tested against these ratios. The most important factor is finding out the value for the False positive rate. This rate defines whether an unsuitable program is recommended to a user. Which can conclude that implicit negative feedback might influence recommendations positively. Comparing different algorithms (e.g. scenarios) might show that one algorithm dominates the other.

6.1 Scenario 1: Liked (viewed) programs

In this scenario a clear distinction was made regarding selecting a program as input for the recommendation process. Whenever the user watched a program, even if it only watched it for 1 minute, the program was selected.

Within this scenario the following steps were followed:

1. A user is selected within the user log database
2. The user profile was cut in half to create a training set and a testset
3. Every program that was watched for longer than 0 minutes was selected
4. The selected programs were been passed to the pattern calculation process
5. Each pattern connected program was compared against the testset

To investigate whether implicit feedback influence the outcome of the recommendations, also the programs that the user did not watched were taken into account. Whenever the negative feedback was used to generate recommendations the following precision and recall ratios were collected:

- **Precision 0.564**
- **Recall 0.19**
- **False Positive Rate 0.482**

These results showed a couple of interesting findings:

- The precision value is higher than the recall value. This is because a filter mechanism is applied to sort the results.
- There is a high false-positive rate which indicates that the implicit feedback has a high influence on the recommendations. As the goal of our recommendations is to recommend a set of good items and not to recommend all good items. A user does not want to cope with a large set of options and therefore a higher precision value is needed to help the user processing these large set of information.

In the following scenario a different approach is presented in selecting programs.

6.2 Scenario 2: Weighted liked programs

This scenario is quite the same as scenario 1 but instead of selecting every program that a user watched for longer than zero seconds, only the programs that a user really watched is selected. To define really a selection mechanism was created to collect only the viewed programs.

Whenever a user watched a program for 1 minute it is likely that this user did not watched this program. He or she might have selected this program but directly after the video has started the user watch another program. To make sure that those programs are not selected, a selection mechanism is created. The selection mechanism execute the following steps in order to select a program within the user profile of a user:

1. A user is selected within the user log database
2. The user profile was cut in half to create a training set and a testset
3. For every program a call is been made to the BBC to gather information about the program. Within the gathered information about the program a filter is been applied to retrieve the duration of the program.
4. If the viewing time, which is stored in the user log, is 10% of the total duration of a program then the program is selected. Several thresholds were tested to find out which threshold was the most optimal. It turned out that most of the recommendations were generated based on the programs that the user watched for several minutes (at least 10% of the total duration). Whenever the threshold was increased the amount of recommendations started to decrease.
5. Those programs are used to generate patterns and were compared against the testset.

The following results are collected after running this scenario on the sample size set:

- **Precision 0.568**
- **Recall 0.20**
- **False Positive Rate 0.502**

Within this scenario similar findings can be concluded. Precision rate is still high and the false positive rate is also quite high. In most cases having such a high false positive rate is not satisfiable. Due to the fact that unwanted results are presented to the user. In case of LD patterns recommendations regarding recommending BBC programs is this value desirable.

6.3 Comparison of scenario 1 and 2

Comparing both scenarios, with respect to the precision-recall ratio, we found that in both cases the false positive rate is quite high. Which indicates that using implicit negative feedback can influence the outcome of the recommendations in positive way. Nevertheless, it is important to note that the recommendations process described in this paper might suite this domain very well. It another domain this might not work or results in an unexpected outcome. Therefore this outcome is preferable within our domain.

Comparing scenario 1 with 2 also led to an interesting finding that both algorithms are not dominating each other. Using a weighted approach of selecting which program should be categorized as a liked program, does increase the false positive rate and recall rate. The increasing value for the false positive rate is due to the fact there are a couple of cases where patterns were found between programs where the user only watched that program for less than 10% of the duration of the program.

Another important finding is that the precision rate is also quite high. In most of the cases the recommendations did satisfy the user profiles. Of course there is still some improvement available, more about this in the discussion/future work sections.

The analysis of those scenarios showed that implicit negative feedback indeed influence TV recommendations created with LD. In the following section a more general analysis is been conducted to find out how many users were reached whenever implicit negative feedback was included and when not.

6.4 General Analysis: scenario 1: Liked (viewed) programs

Within this scenario the data for each user was been separated in two parts: a training set and a testset. Based on the training set the programs that were viewed longer than the given threshold were stored and used for the recommendations. The programs are used as input for our pattern calculating process and the results of that are ranked via the ranking algorithm. The ranked results were compared against the testset and the results of that are:

Number of users reached		
	Number of users	%
BBC One	492354	7,65
BBC Two	388269	6,03
BBC Three	164030	2,54
BBC Four	90876	1,41
BBC News	356251	5,53
BBC Parliament	24908	0,38

Table 8: Overview of how many users were reached via our TV pattern recommendations

In table 8 an overview of the amount of users that were reached through our pattern recommendations, is given. We compared the results with the BBC user logs and we found that we generated for 23,54% of the users preferable recommendations. For all these recommendations we found that for 87% of the cases the ranked Top 5 was within the testset of a user. The percentages were calculated based on the total number of users that were logged within the database, within a three month timespan: 6434924 users. This value represents all the users that are stored within the database. In addition, this also includes all the users were not recommendations were generated for.

6.5 General Analysis: Scenario 2: Liked and disliked programs

In this scenario not only the programs that the user actually watched is been taken into account but also the programs that the user did not watched (e.g. implicit negative feedback). With this scenario we wanted to see that if we also took the not viewed programs as input for our recommendation process then we might reach a larger group of users.

To achieve this scenario we had to create a classification method which made it possible to use all the programs that were available within the training set of each user. To classify the results we had to add weights to the recommendations. For each program that the user watched we classified the programs that were connected through patterns with the value = 1. For all the programs that the user did not watched we classified them with the value = 0. If in the end a program is recommended through a pattern and this program is within the testset of the user, the weighted value will be added up with a value of 0.5. We compared the recommendation results against the testset and the following results were found:

Number of users reached		
	Number of users	%
BBC One	512154	7,95
BBC Two	399069	6,20
BBC Three	169030	2,62
BBC Four	93876	1,45
BBC News	391237	6,07
BBC Parliament	25258	0,39

Table 9: Overview of how much users were reached via our TV pattern recommendations

In table 9 an overview of the results of scenario two are visible. Interesting to see is that the total amount of users now reached 24,68% of the total amount of users. Which is a difference of 1,14%.

Compared to the first scenario we see a small improvement. In contrast, the computational time to calculate this scenario is exponentially bigger.

6.6 Comparison of scenarios

To compare the outcome of the two scenarios a Student's T-test[19] is conducted to test the null hypothesis: "Using not viewed and viewed programs as input, influence the LD pattern recommendations more". The desired result is that comparing the two scenarios one method will reject the null hypothesis and performs better than the other.

As input for the T-test we compared two groups (two scenarios) where the total amount of users per channel is used as individual input. The data for the two groups could be defined as:

- Group A = {492354,388269,164030,90876,356251,24908}
- Group B = {512154,399069,169030,93876,391237,25258}

Both data lists were used as input for the Student's T-test and the following results were, visible within table 10 & 11:

Calculating the Student's T-test resulted in a p-value of 0.810[21]. With a p-value of 0.810 the null hypothesis is not rejected. The differences between the two scenarios is not large enough to distinguish which method is better than the other.

[21]Calculated via `http://www.ruf.rice.edu/~bioslabs/tools/stats/ttest.html` and Excel function ttest

Group A: t-test results	
Mean	252800
95% confidence interval	from 79369 till 426190
Standard Deviation	185700
Hi	492400
Low	24910
Median	260100

Table 10: T-test result for Group A (Scenario 1)

Group B: t-test results	
Mean	265100
95% confidence interval	from 91692 thru 438520
Standard Deviation	195400
Hi	512200
Low	25260
Median	280100

Table 11: T-test result for Group B (Scenario 2)

Hence, doing this experiment and investigating the patterns and recommendations the following remarks could be made:

- **BBC News** Most of the patterns that we created for the programs, that were aired on the BBC News channel, also resulted in other news programs. A small set of programs did actually connect to other BBC programs which were not news related. However, that situation did not occured in the testset.
- **BBC Parliament** A similar image is visible with this channel. Every user that watched a BBC Parliament program almost ended up in watching another program from the same channel. There was also a clear connection visible between the BBC Parliament channel and the BBC News channel. Subsequently, during an election period this channel will be viewed more often than not.
- **Others** With all the other channels we see no specific pattern through channels. Programs from the BBC One are connected towards all other BBC Channels and vice versa.
- **Popularity** A clear difference can be found within the popularity on Twitter between different channels. In general, the programs that were aired on the BBC One were more popular on Twitter than other channels. That does not alter that other programs on other channels might be less popular. The BBC often aired a specific program on the BBC One and after a certain timespan other channels are showing the same program again.
- **Ranking** By adding the WR measurement the programs that were strongly connected with other programs, due to the large amount of patterns that they share, did not always ended up high in the ranking. The popularity score influenced in the end the outcome of the ranking. Despite this, a comparison is made between ranking only based on the total number of patterns and ranking based on the number of patterns and added popularity score. The ranking of the

recommendations were slightly adjusted by adding the popularity score and the calculated WR score. Thus, the recommendations were generated together with the popularity score and filtered based on the WR.

7. DISCUSSION

It has been shown that using implicit negative feedback together with the created TV pattern recommendations can influence the results. The total reach that we achieved was a quarter of the total amount of users. Nevertheless, there is still room for improvements and there are a few limitations which are explained in the following subsections.

7.1 Limitations

Although we found interesting results during this research, there were a few limitations that we encountered.

First, generating a recommendation list for each user within the userlog is a time consuming process. Running a script against a 2GB large database takes a large amount of time, especially when a lot of filtering needed. To give an idea: generating recommendations for each user, within a three months timespan, costed approximately one whole week to run. Excluding the amount of time that is needed to generate the patterns. Nevertheless, this last process only needs to be done once.

Second, generating a social media popularity score is hard. Often we encountered programs that did not returned a proper result on our Twitter API call. Which could indicate that the program is not very popular on the social media but perhaps other social media tools could improve the popularity score of the programs.

Third, generating a social media popularity score is not always available. Often we encountered programs that did not returned a proper result on our Twitter API call. Which could indicate that the program is not very popular on the social media but perhaps other social media tools could improve the popularity score of the programs.

Fifth, several parameters can be optimized to increase or decrease the relevance of the recommendations towards a user. Parameters as the ratio between training and test set, the liked/disliked (viewing time) ratio and the presented recommendation list value can influence the recommendations significantly. Playing with these values is a time consuming process because whenever these values might work properly within a small sample set it might not work very well within the whole dataset. Therefore a more in depth research should be conducted to find out which parameter settings are the optimal with respect to this results.

Finally, we found that a large set of the patterns were connected to CBEEBIES and CBBC programs. We decided not to use these programs due to the fact that we do not have access to watch children's programs from outside the United Kingdom. In conclusion, if we want to create and interactive online application (which we labeled as future work) we need to filter out these CBEEBIES and CBBC programs.

7.2 Future Work

During the research and implementation of the LD recommender we found interesting features and improvements which could be added into future releases.

First, we have tested the recommendations via offline evaluations. The BBC provided us user logs which gave us insight in what users watched within a certain timeframe. To actually test this within a live environment would be a good indicator whether the recommendations not only works on offline data but also with live data. An implementation within an online environment would be the next step.

Second, the testset was defined per user. If a user has ten entries within the database the testset consisted out of five programs. Perhaps adjusting the size of the training set and/or testset might influence the overal performance of the TV recommender.

Third, another important factor is conducting a proper social media study. Due to time constraints a proper social media study is not conducted. We used Twitter to calculate the popularity score but perhaps there are other methods available that could extend and/or improve the current popularity score, perhaps a combination of different social media platforms is the way to go.

Finally, filtering the generated patterns on CBeebies programs is needed to recommend relative TV programs. It is now often the case that a CBeebies program is connected to a BBC One or BBC Two program. We could label this as an unexpected result which might interest the user but we can only check this via online evaluations.

8. CONCLUSION

Creating a LD recommender to recommend TV programs to users and doing research on the influence of implicit negative feedback on TV recommendations was the main goal of this paper. We created an recommender which uses semantic enriched metadata to solve the problem of sparsity. We added several methods for calculating the rank of the recommendations. One of the methods was calculating the popularity score of the programs. We tried to increase the relevance by using social media to find out how popular a specific program is on the web. After doing experiments and testing two scenarios we found out that implicit negative feedback indeed influence the recommendations. The false positive rate showed that for a large group of users the recommendations were relevant.

Determine whether the influence of LD is high or low is a hard job to do. Different testing methods are needed to actually say something about the influence. Furthermore, to extend this research and to find out if the influence of LD is higher compared to a traditional content-based or collaborative recommender a comparison study needs to be conducted. That aside, the outcome of the patterns and the recommended TV programs might not cover the whole dataset and satisfy all the users within this dataset, it is still a possibility that users might like the recommended result. To test this an online study should be conducted to find out if those unexpected results are satisfying the needs of the users.

To conclude, the process of generating patterns for recommendations is as interesting approach. It will recommend possible new items which might never come to the surface compared to traditional recommendation systems. The outcome of this research can be used as base for further development and research.

9. REFERENCES

[1] X. Amatriain and J. Basilico. Netflix recommendations: Beyond the 5 stars (part 1)–the

netflix tech blog. *URL http://techblog. netflix. com/2012/04/netflix-recommendations-beyond-5-stars. html*, 2012.

[2] D. Asanov. Algorithms and methods in recommender systems. *Berlin Institute of Technology, Berlin, Germany*, 2011.

[3] A. Bellogín, I. Cantador, F. Díez, P. Castells, and E. Chavarriaga. An empirical comparison of social, collaborative filtering, and hybrid recommenders. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(1):14, 2013.

[4] C. Bizer, T. Heath, K. Idehen, and T. Berners-Lee. Linked data on the web (ldow2008). In *Proceedings of the 17th international conference on World Wide Web*, pages 1265–1266. ACM, 2008.

[5] R. Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[6] A. Gunawardana and G. Shani. A survey of accuracy evaluation metrics of recommendation tasks. *The Journal of Machine Learning Research*, 10:2935–2962, 2009.

[7] W. Hill, L. Stead, M. Rosenstein, and G. Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201. ACM Press/Addison-Wesley Publishing Co., 1995.

[8] Z. Huang, H. Chen, and D. Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1):116–142, 2004.

[9] D. Kelly and J. Teevan. Implicit feedback for inferring user preference: a bibliography. In *ACM SIGIR Forum*, volume 37, pages 18–28. ACM, 2003.

[10] G. Linden, B. Smith, and J. York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[11] G. D. Linden, J. A. Jacobi, and E. A. Benson. Collaborative recommendations using item-to-item similarity mappings, July 24 2001. US Patent 6,266,649.

[12] V. Maccatrozzo, L. Aroyo, G. Schreiber, L. Miller, and C. Newell. Improving content-based recommendations with semantic enrichment: the infinite trailers case study. *Journal of Web Semantics*, 2015.

[13] V. Maccatrozzo, L. Aroyo, W. R. Van Hage, et al. Crowdsourced evaluation of semantic patterns for recommendation. In *UMAP Workshops*, 2013.

[14] V. Maccatrozzo, D. Ceolin, L. Aroyo, and P. Groth. Semantic pattern-based recommender. *Springer, Heidelberg*, 475:182–187, 2014.

[15] E. Pariser. *The filter bubble: What the Internet is hiding from you.* Penguin UK, 2011.

[16] A. Passant. dbrec–music recommendations using dbpedia. In *The Semantic Web–ISWC 2010*, pages 209–224. Springer, 2010.

[17] M. J. Pazzani and D. Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[18] F. Ricci, L. Rokach, and B. Shapira. *Introduction to recommender systems handbook.* Springer, 2011.

[19] W. R. Rice. Analyzing tables of statistical tests. *Evolution*, pages 223–225, 1989.

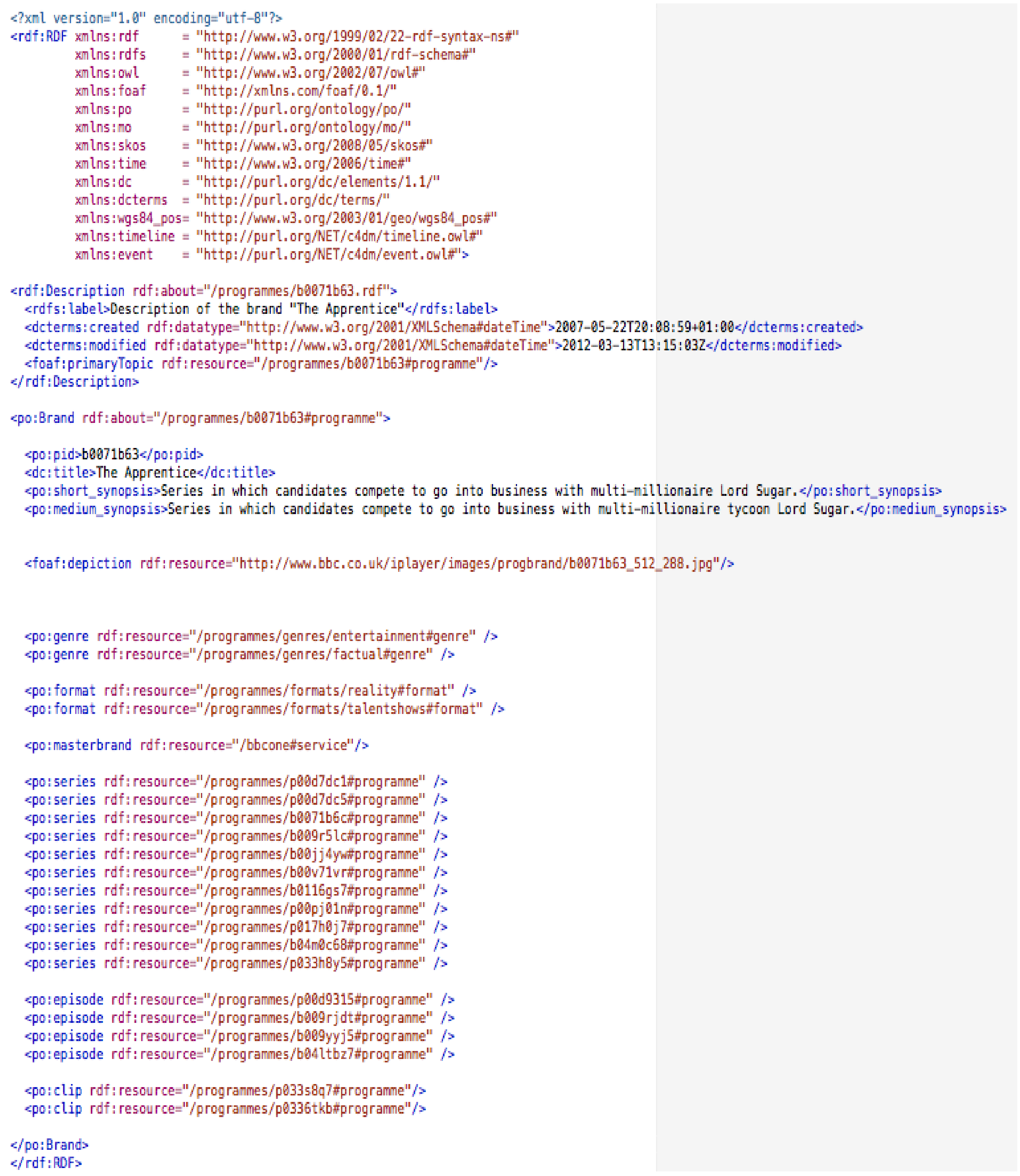
[20] G. Rizzo and R. Troncy. Nerd: a framework for unifying named entity recognition and disambiguation extraction tools. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 73–76. Association for Computational Linguistics, 2012.

[21] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[22] B. Schopman, D. Brickly, L. Aroyo, C. van Aart, V. Buser, R. Siebes, L. Nixon, L. Miller, V. Malaise, M. Minno, et al. Notube: making the web part of personalised tv. 2010.

APPENDIX

A. A STRUCTURED DATA OVERVIEW OF A BBC PROGRAM



```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:rdf      = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs     = "http://www.w3.org/2000/01/rdf-schema#"
         xmlns:owl      = "http://www.w3.org/2002/07/owl#"
         xmlns:foaf     = "http://xmlns.com/foaf/0.1/"
         xmlns:po       = "http://purl.org/ontology/po/"
         xmlns:mo       = "http://purl.org/ontology/mo/"
         xmlns:skos     = "http://www.w3.org/2008/05/skos#"
         xmlns:time     = "http://www.w3.org/2006/time#"
         xmlns:dc       = "http://purl.org/dc/elements/1.1/"
         xmlns:dcterms  = "http://purl.org/dc/terms/"
         xmlns:wgs84_pos= "http://www.w3.org/2003/01/geo/wgs84_pos#"
         xmlns:timeline = "http://purl.org/NET/c4dm/timeline.owl#"
         xmlns:event    = "http://purl.org/NET/c4dm/event.owl#">

<rdf:Description rdf:about="/programmes/b0071b63.rdf">
  <rdfs:label>Description of the brand "The Apprentice"</rdfs:label>
  <dcterms:created rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2007-05-22T20:08:59+01:00</dcterms:created>
  <dcterms:modified rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2012-03-13T13:15:03Z</dcterms:modified>
  <foaf:primaryTopic rdf:resource="/programmes/b0071b63#programme"/>
</rdf:Description>

<po:Brand rdf:about="/programmes/b0071b63#programme">

  <po:pid>b0071b63</po:pid>
  <dc:title>The Apprentice</dc:title>
  <po:short_synopsis>Series in which candidates compete to go into business with multi-millionaire Lord Sugar.</po:short_synopsis>
  <po:medium_synopsis>Series in which candidates compete to go into business with multi-millionaire tycoon Lord Sugar.</po:medium_synopsis>


  <foaf:depiction rdf:resource="http://www.bbc.co.uk/iplayer/images/progbrand/b0071b63_512_288.jpg"/>



  <po:genre rdf:resource="/programmes/genres/entertainment#genre" />
  <po:genre rdf:resource="/programmes/genres/factual#genre" />

  <po:format rdf:resource="/programmes/formats/reality#format" />
  <po:format rdf:resource="/programmes/formats/talentshows#format" />

  <po:masterbrand rdf:resource="/bbcone#service"/>

  <po:series rdf:resource="/programmes/p00d7dc1#programme" />
  <po:series rdf:resource="/programmes/p00d7dc5#programme" />
  <po:series rdf:resource="/programmes/b0071b6c#programme" />
  <po:series rdf:resource="/programmes/b009r5lc#programme" />
  <po:series rdf:resource="/programmes/b00jj4yw#programme" />
  <po:series rdf:resource="/programmes/b00v71vr#programme" />
  <po:series rdf:resource="/programmes/b0116gs7#programme" />
  <po:series rdf:resource="/programmes/p00pj01n#programme" />
  <po:series rdf:resource="/programmes/p017h0j7#programme" />
  <po:series rdf:resource="/programmes/b04m0c68#programme" />
  <po:series rdf:resource="/programmes/p033h8y5#programme" />

  <po:episode rdf:resource="/programmes/p00d9315#programme" />
  <po:episode rdf:resource="/programmes/b009rjdt#programme" />
  <po:episode rdf:resource="/programmes/b009yyj5#programme" />
  <po:episode rdf:resource="/programmes/b04ltbz7#programme" />

  <po:clip rdf:resource="/programmes/p033s8q7#programme"/>
  <po:clip rdf:resource="/programmes/p0336tkb#programme"/>

</po:Brand>
</rdf:RDF>
```

Figure 5: A RDF example of the BBC programme: The apprentice

B. SPARQL QUERY

SPARQL query to retrieve all the unique programs that are used for generating patterns.

```
[!htbp]
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT ?sub WHERE {
   ?sub ?pred ?obj
   FILTER ( regex(str(?sub),"http://www.bbc.co.uk/programmes"))
}
```

C. ALGORITHMS

```
Algorithm 1 STEP 1: Retrieve User Data from Database (DB)
Require: Loading the user logs into the DB, querying the DB to find a specific
    user. Split per user the total data field in half and store them in two parts:
    training set and a testset
 1: for each user do
 2:    if viewingtime in trainingset is > 0 then
 3:       Store program and userid as userviewedprogram
 4:    else if viewingtime in trainingset is == 0 then
 5:       Store program and userid as usernotviewedprogram
 6:    end if
 7: end for
 8: return userviewedprogram, usernotviewedprogram
```

```
Algorithm 2 STEP 2: Retrieve TV Patterns from API
Require: The TV Pattern API
 1: for each user do
 2:    for i in range(0,userviewedprogram) do
 3:       for j in range(0,userviewedprogram do
 4:          compare userviewedprogram[j] with userviewedprogram[i] via the
            API
 5:          Collect all the patterns and store them in numberofpatterns
 6:          j++
 7:       end for
 8:    end for
 9: end for
10: return userviewedprogram[j], userviewedprogram[i], numberofpatterns
```

```
Algorithm 3 STEP 3: TV Pattern Ranking Algorithm
Require: The input of algorithm 1 and 2, Twitter API
 1: for each user do
 2:    Get the popularityscore per program via the TWITTER API
 3:    Store the number of patterns that a programx shared with a programy
       together with the popularityscore
 4:    Calculate the WeightedRating(WR) score
 5:    Sort the recommendations based on WR score
 6:    Check the sorted recommendations against the testset
 7:    Compare the occurency of recommendations within the testset of that
       user.
 8: end for
 9: return sortedlist[[1,program pid],[2,program pid],. . . ]
```

Figure 6: Pseudocode of the recommendation process