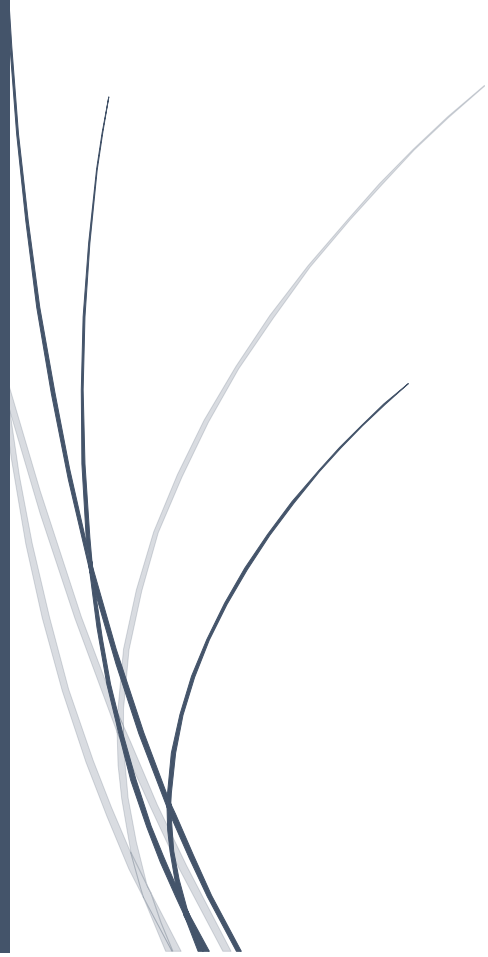
UNIVERSITY OF TWENTE
2014 / 2015

Recommendations using DBpedia

How your Facebook profile can be used to find your next greeting card



MASTER THESIS

Author:

Anne van de Venis

Supervisors:

Maurice van Keulen

Djoerd Hiemstra

Victor de Graaff

Foppe Strikwerda

CONTENTS

List of figures 5

List of tables 6

Abstract 7

Introduction 9

1.1 Background & Motivation 9

1.2 Problem Statement 11

1.3 Research Questions 12

1.4 Proposed System Architecture 12

1.5 Contributions 13

1.6 Document Structure 14

Case Study 15

2.1 The company 15

2.2 Data set 16

2.3 Kaartje2go and IBRS 17

State of the Art 19

3.1 Recommender Systems 19

3.1.1 Collaborative-based 20

3.1.2 Content-based methods 21

3.1.3 Demographic approaches 22

3.1.4 Knowledge-based approaches 23

3.1.5 Hybrid approaches 23

3.1.6 Problems 23

3.2 Semantic Web 24

3.2.1 Relevance search 27

3.3 Existing applications 29

3.3.1 MoviExplain: A Recommender System with Explanations 29
3.3.2 Linked Open Data for content-based recommender systems 29
3.3.3 HeteRecom: A Semantic-based Recommendation System in Heterogeneous Networks 30
3.3.4 dbrec - Music Recommendations Using DBpedia 30
3.3.5 Sprank 31
3.4 Conclusion 32
Technology 33
4.1 Global overview 33
4.2 IBRS 35
4.3 Detailed overview 38
4.3.1 Facebook mapper 38
4.3.2 DBpedia explorer 40
4.3.3 Tag selector 44
4.3.4 Recommendation creator 45
Evaluation 47
5.1 Baseline systems 47
5.2 Website 48
5.3 Results 52
5.3.1 Results with FacebookOnly 52
5.3.2 Results with InvertedIBRS 54
5.4 Validation results 56
Conclusions 59
Acknowledgements 61
Bibliography 63

LIST OF FIGURES

Figure 1: Movie recommendations created by Netflix 10

Figure 2: Overview of the proposed RS 12

Figure 3: Semantic paths from Cristiano Ronaldo to Football Club and Portugal 13

Figure 4: Kaartje2go website 15

Figure 5: Data model of the Postcards and related tags from Kaartje2go 16

Figure 6: Tag distribution for the cards. 17

Figure 7: MAE performance of adjusted cosine, cosine and Pearson correlation. [17] 20

Figure 8: Basic process of demographic generalization. [26] 22

Figure 9: RDF Data model describing a document and the author. [44] 24

Figure 10: Data model described using the Dublin Core vocabulary. [44] 25

Figure 11: Linked Open Data (LOD) cloud on August 30th, 2014. [45] 26

Figure 12: Heterogeneous Information networks. [50] 27

Figure 13: (a) Tensor representation of the RDF graph. (b) Slices decomposition. [53] 29

Figure 14: Interface of dbrec. [56] 31

Figure 15: Global overview of IBRS. 34

Figure 16(a): Facebook page about Cristiano Ronaldo leads to the tags Madrid, Portugal, Lissabon and Football. 34

Figure 17: Schematic overview of the database used in IBRS. 36

Figure 18: Facebook likes map results for the Dutch and English DBpedia dataset. 40

Figure 19: Four different ways of selecting related items C via a middle node B, where related items are found using (a) only outlinks, (b) outlinks from node A and inlinks from node B, (c) inlinks from A and outlinks from B and (d) only using inlinks. 41

Figure 20: (a) Direct relation from The Beatles (b) The Beatles moving towards broader concepts. 41

Figure 21: Welcome screen of the evaluation website. 48

Figure 22: Screenshot of evaluation page for cards. 49

Figure 23: Screenshot for the evaluation of the recommendations for holiday homes. 50

Figure 24: Screenshot for the evaluation of the recommendations for holiday homes using the Likert scale. 51

Figure 25: Overview of the results for the cards for the two evaluation pages. 52

Figure 26: Overview of the results for the holiday houses for the two evaluation pages. 53

Figure 27: Votes results for ibrs vs invertedibrs 54
Figure 28: Vote results with most votes per user. 55
Figure 29: Votes results with rating of recommendations. 56

LIST OF TABLES

Table 1: 20 most used tags for the postcards 17
Table 2: A Fragment of a Rating Matrix for a Movie Recommender System [14] 20
Table 3: Server configuration settings 37
Table 4: Meta information that is added to the URI's 38
Table 5: Used Sparql query 39
Table 6: Related items results for Cristiano Ronaldo. 42
Table 7: Found related items for Amsterdam. 42
Table 8: Related items for Arctic Monkeys. 43
Table 9: Cleaned top 20 tags. 44
Table 10: Found tags. Translated from Dutch for the reader's convenience. 45

ABSTRACT

Recommender systems (RS) are systems that provide suggestions that users may find interesting. In this thesis we present our Interest-Based Recommender System (IBRS) that can recommend tagged item sets from any domain. This RS is validated with item sets from two different domains, namely postcards and holidays homes. While postcards and holiday homes are very different items, with different characteristics, IBRS uses the same recommender engine to create recommendations.

IBRS solves several problems that are present in classic RSs, such as the cold-start problem and language independence. The cold-start problem for new users, is solved by using Facebook likes for creating a user profile. It uses information in DBpedia to create recommendations in a tag-based item set for multiple domains, independent of the language. Using both external knowledge sources and user content, makes our system a hybrid of a knowledge-based and content-based RS.

We validated our system through an online evaluation system in two evaluation rounds with test user groups of approximately 71 and 44 people.

The main contributions in this thesis are:

- a literature study of existing recommendation approaches;
- a language-independent mapping approach for tags and social media resource onto DBpedia resources;
- a domain-independent algorithm for detecting related concepts in the DBpedia graph;
- a recommendation approach based on both Facebook and DBpedia;
- a validation of our recommendation approach.

CHAPTER 1:

INTRODUCTION

How can our Facebook-likes help us to find which greeting cards we will send to our friends for our next birthday? Finding our way through the massive amount of information on the web is a daily struggle for many internet users. More and more websites rely on user profiling to improve the user experience, and help the user to find what he is looking for efficiently. In this thesis, we attempt to assist users in their struggle by detecting his interests from his Facebook profile. We use additional knowledge from the web to create a match between the user and items in a product database, such as greeting cards, holiday homes, or anything else.

The rest of this chapter is structured as follows: Section 1.1 contains the background and motivation for this research, Section 1.2 describes the problem statement, Section 1.3 contains the research questions, Section 1.4 describes the proposed system architecture, Section 1.5 contains the contributions of this thesis and Section 1.6 describes how the rest of this thesis is structured.

1.1 BACKGROUND & MOTIVATION

Customers are buying more and more products online via web shops. E-commerce sales in the Netherlands have increased with 42% over the last 8 years [1]. Online shops have a lot of advantages, compared to real shops, like:

a. access from anywhere, at any time;
b. online reviews, and;
c. a wide variety of choices.

The latter point is one of biggest advantages of e-commerce, but it can also overwhelm customers. The challenge therefore is to provide an interface that enables customers to find a greeting card of their choice within their attention span. This problem is present in most online shops, since most web shops have so many products that it is very hard for customers to select the set of products that they are interested in.

Recommender systems have been around for decades to help customers select products. Different types of a Recommender System (RS) have been researched and developed and are widely implemented, like for example in [2] [3] [4].

Current RSs, such as the one Netflix uses for movie and TV show recommendation, as illustrated in Figure 1, perform well in providing useful product recommendations to the customers. In 2012, Netflix presented results of their RS: 75% of the items were picked based on the recommendations their RS gave [5]. This shows the recommendations have a high impact on the behavior of the customer. Companies, like Netflix, invest a lot of money in improving their RSs. Netflix even launched the “Netflix Prize” in 2006 [6]. They offered 1 million dollars for the algorithm that could improve the performance of their existing system by 10%.

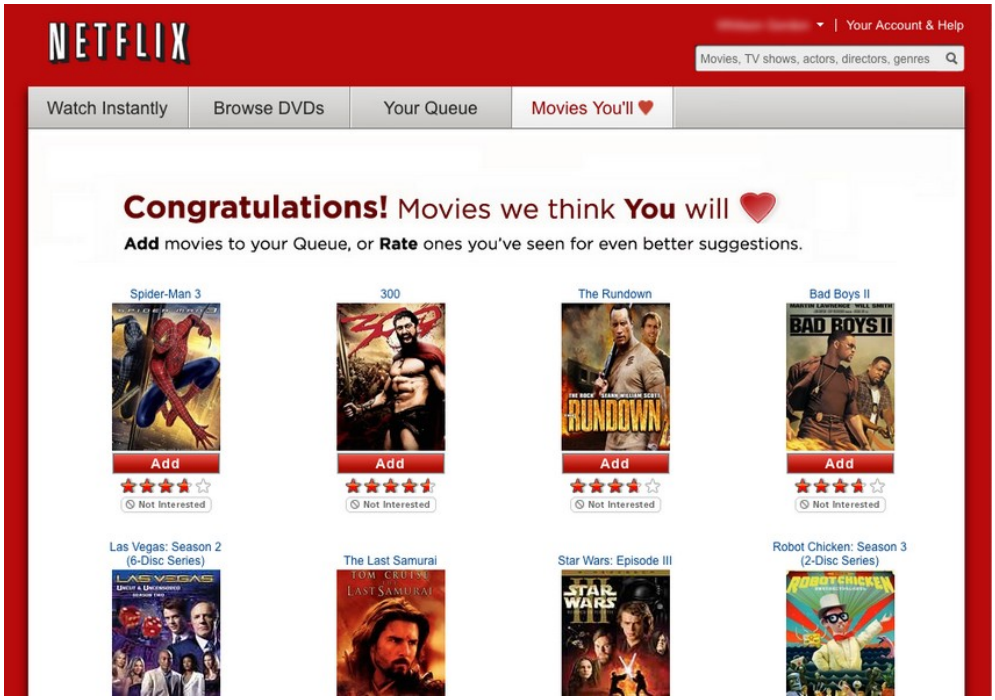


FIGURE 1: MOVIE RECOMMENDATIONS CREATED BY NETFLIX

There exist several techniques to create RSs. Some well-known techniques include content-based or collaborative filtering-based RSs, which use the user's feedback to improve the ranking mechanism, as described by Burke [7]. This feedback could be implicit, like giving ratings to items or implicit feedback like the scrolling behavior on a webpage of a product. The performance of these systems improve as more users give feedback.

In this thesis, we present a novel, semantic web-based method for finding interesting related items that works in all domains.

1.2 PROBLEM STATEMENT

In this work, we investigate the possibility to create a RS that can give recommendations based on a user's social profile. It should be able to create recommendations from tagged item sets in different domains.

Several different types of RSs have been designed to created recommendations for users, based on very different principles. Still, after decades of experience it is hard to implement a good RS.

Cold-start problem

Bobadilla et al [8] describe a commonly encountered problem known as the 'cold-start problem'. There exist RSs that are based on historical data like ratings of users or product features. This data is used by RS to create a profile for user *u* based on similar users or items. Therefore, these systems are not very useful in new applications.

Domain-independence

Content-based RSs also have another shortcoming, since they are limited in content analysis [9]. They often need specific domain knowledge and are therefore designed and trained to work in a specific domain. They cannot create recommendations in different domains nor create cross-domain recommendations.

Synonyms

The items from social profiles are often instances of broader concepts. For example, "Cristiano Ronaldo" is related to "Football" or "Soccer". The tags in tag-based RSs often contain these broader concepts. "Football" and "Soccer" are different tags but are in some regions used for the same concept. As described by Zanardi and Capra [10] these synonyms are a well-known problem in tag-based RSs.

1.3 RESEARCH QUESTIONS

This section gives an overview of the research questions that are needed to answer to problem statement.

- RQ-1 - What is the current state of the art in recommender systems?
- RQ-2 - How can we find related items for user's interests?
- RQ-3 - How can we extract recommendations based on semantic relations?

1.4 PROPOSED SYSTEM ARCHITECTURE

Our system shall overcome the cold-start problem using personal information available in social media. A research by NewCon [11] state that 87% of the users between the 20-39 years old use Facebook. Since Facebook has a good API for fetching a user's profile and it have a lot of users, it is a good place to start creating profiles for our RS.

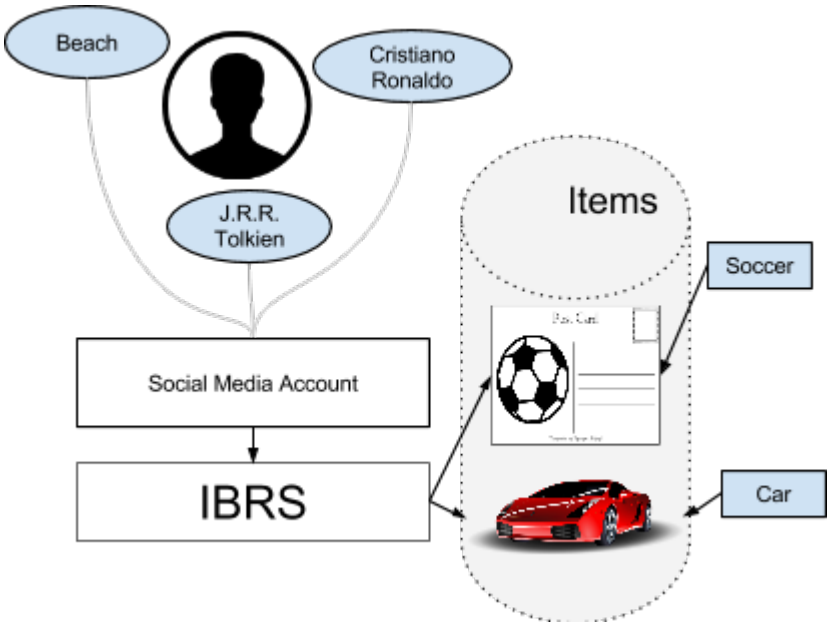


FIGURE 2: OVERVIEW OF THE PROPOSED RS

The RS contains items that are associated with tags, these tags will be matched to the likes of the user. Matched tags are used to create recommendations. This process is illustrated in Figure 2.

As explained in Section 1.2, tags can contain very general concepts. Items from a user's profile may contain instances of these general concepts. In this thesis we present an approach to match these instances to the broader concepts, using paths, so they can be mapped to tags. In Figure 3 we show how football player Cristiano Ronaldo can be mapped to "*football club*" and "*Portugal*".

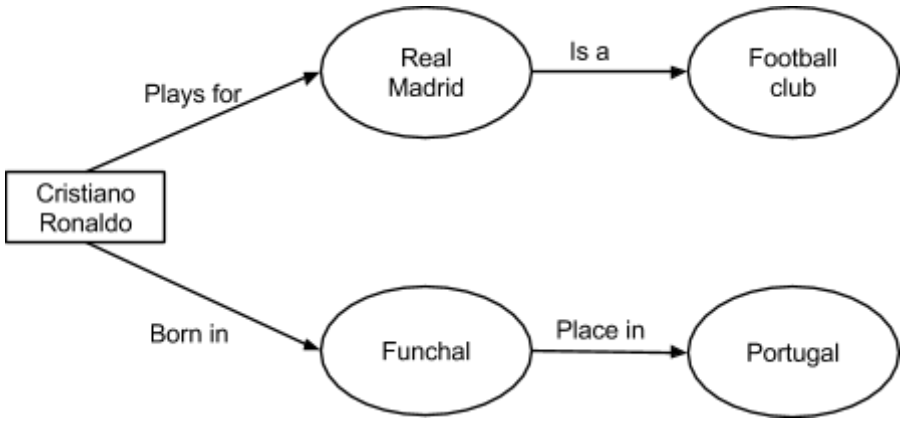


FIGURE 3: SEMANTIC PATHS FROM CRISTIANO RONALDO TO FOOTBALL CLUB AND PORTUGAL

Since the RS creates recommendations based on the user's interests, the RS proposed in this thesis is called *Interest-Based Recommender System* (IBRS). Since it is based on tags, IBRS can be used in different domains. It can create a recommendation for a user *u* based on his set of interests. The main principle of IBRS is that users are more interested in items that are closely related to items we know they like.

1.5 CONTRIBUTIONS

Our contributions to the research in the field of semantic-based RSs consist of the following items:

- an overview of existing recommendations approaches;
- a language-independent mapping approach for tags and social media resources onto DBpedia resources;
- a domain-independent algorithm for detection of related concepts in the DBpedia graph;
- an approach to create recommendations using both Facebook and DBpedia;
- a validation of our recommendation approach.

1.6 DOCUMENT STRUCTURE

This thesis is further structured as follows: Chapter 2 describes the case study used in this thesis. Chapter 3 contains the state of the art in RSs and semantic web, based on a literature study. Chapter 4 describes how the Semantic Web can be used to find related content and how this theory can be used in a real application that recommends items. Chapter 5 describes the results and evaluation of the RS. Chapter 6 gives a final conclusion and pointers to possible future work.

CHAPTER 2:

CASE STUDY

Throughout this thesis, we use the case of Kaartje2go, an online portal for offline greeting cards, as a running example. We also used their product database as one of our validation sets. In this chapter we describe what Kaartje2go is, and how they contributed in this research.

2.1 THE COMPANY

In this thesis, we use Kaartje2go as our running example for a web shop. Kaartje2go is an online portal for greeting cards, with over 986,000 registered customers. Customers can create postcards from more than 47,000 templates or start from scratch. The greeting cards that people order, are shipped to their family and friends in print directly, potentially with a personal message of the sender. Since the start of Kaartje2go in 2006, their customers have sent over 20 million postcards. Over the years, Kaartje2go has won several customer awards. Figure 4shows a screenshot of the homepage.

FIGURE 4: KAARTJE2GO WEBSITE

Kaartje2go offers two sending methods, namely:

- 1:1 (one-to-one): the customer sends his card to one address. These cards are often personal and meant for one person or family.
- 1:n (one-to-many): the customer sends a single card to multiple addresses. These cards are often less personal and used for invitations, best wishes and holiday cards. The cards can be sent directly to all receivers directly or in a box to the customer.

One-to-many cards are created based on the taste of the sender. However, one-to-one cards are often created based on the taste of the receiver. One-to-many deliveries is by far the most selected delivery method. More than 75% of all purchases are sent in a box, containing a set of one-to-many cards. This means that most sold cards are based on the taste of the sender. The research described in this thesis focuses on the RS user, and thus in assisting customers to find interesting cards from the sender-perspective.

While customers send lots of postcards, they sometimes find it difficult to find the right postcard. Usage statistics show that the internal search function does not suit the need of the customer. For instance, the search result page for the search query 'funny' has an exit ratio of 54%. Considering that this is the most popular query right now, with 12.000 searches in the last year, Kaartje2go is looking for a way to improve their search functionality, with the ultimate goal to improve their sales-to-visit ratio.

2.2 DATA SET

In this thesis we used the complete set of postcards and related tags from Kaartje2go. This dataset was delivered as a MySQL database dump. A graphical overview of this dataset can be found in Figure 5.

FIGURE 5: DATA MODEL OF THE POSTCARDS AND RELATED TAGS FROM KAARTJE2GO

This dataset contains 46,338 postcards and 13,571 unique tags. On average, every postcard is related to 5.04 tags. More than 23,000 postcards are related to five tags, as seen in the complete tag distribution in Figure 6.

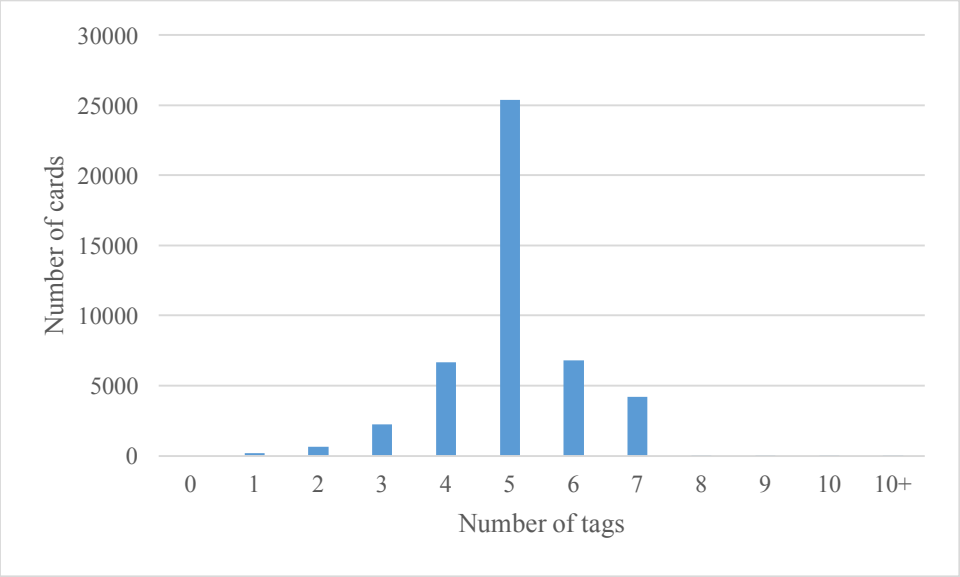


FIGURE 6: TAG DISTRIBUTION FOR THE CARDS.

Table 1 gives an overview of the top-20 most used tags in this dataset. These tags are translated from Dutch for the reader's convenience. These most used tags do not describe the content of the postcards, but mainly the category or sentiment of the postcard.

Tag	# postcards	Tag	# postcards
congratulation	7,564	just	3,309
birthday	7,534	sweet	2,810
birth	4,671	girl	2,221
invitation card	4,500	love	2,136
cheerful	4,378	baby	2,056
Christmas	3,874	boy	2,039
party	3,761	hearts	1,921
flowers	3,730	birthday girl	1,839
anniversary	3,465	photo	1,793
congratulations	3,328	birthday boy	1,774

TABLE 1: 20 MOST USED TAGS FOR THE POSTCARDS

2.3 KAARTJE2GO AND IBRS

As we will show in Chapter 4, the tags provided by the Kaartje2go dataset are very suitable for our recommender system IBRS. However, the dataset contains lots of less useful duplicated tags, like singular and plural or Dutch and English tags of the same concept. For example the tags ‘congratulation’ and ‘congratulations’ from Table 1 describe the same concept. IBRS should not care about the form of the tag to create the recommendations.

The first step will be to clean the tags by mapping them to structured information available on the web. More structured information will be used to related greeting cards using the tags.

CHAPTER 3:

STATE OF THE ART

This chapter describes the results from the literature study that was conducted for this thesis. In this work, we aim to build a RS that is based on the semantic web. Therefore, we first discuss the work done in the RS field in general in Section 2.1. Then we discuss the semantic web in Section 2.2. In Section 2.3 finally, an overview is given of those RS solutions that are based on the semantic web.

3.1 RECOMMENDER SYSTEMS

In recent years a lot of work has been done in Recommender System research.

Ricco et al. describe RSs using three objects, namely users, items and transactions [12].

- *Users*
  A *user* is the person a recommendation is intended for. The RS creates recommendations based on a user profile. This user profile can be socio-demographic data, such as age, gender and education or based on the user's behavior.
- *Items*
  An *item* is an object that can be recommended to users. Examples include, books, CD's, holidays and postcards. They can be represented as a single value, like a name or an id, or they contain various attributes that can be used in the recommendation process.
- *Transactions*
  A *transaction* is the recorded interaction between the user and the RS. Based on the transactions the RS can build a profile of a user. Pazzani and Billsus described different feedback methods in [13] that can be used to construct a profile of a user, namely using explicit and/or implicit feedback. Explicit feedback indicates the relevance of an item for a user. Popular feedback mechanisms are for example numbers, letters, stars, hearts, etc. Implicit feedback can be obtained from the behavior of a user, like click actions and time spent viewing the item. Most RSs are based on explicit feedback based on ratings. Therefore, we will use the terms transactions and ratings interchangeably in this paper.

The purpose of a RS is to predict future transactions between users and items. Based on the estimated transactions, a list of recommendations can be constructed.

Adomavicius et al. [14] give a formal formulation for the recommendation problem, in terms of C, the set of all users (customers) and S, the set of all items that can be recommended. To capture the usefulness of an item s for user u they define a utility function u: $C \times S \rightarrow R$, where R is an ordered list of integer ratings. The recommendation problem is that not u is not defined for the whole $C \times S$ space, but only for a small subset like for example the rating matrix in Table 2.

	K-Pax	Life of Brian	Memento	Notorious
Alice	4	3	2	4
Bob	-	4	5	5
Cindy	2	2	4	-
David	3	-	5	2

TABLE 2: A FRAGMENT OF A RATING MATRIX FOR A MOVIE RECOMMENDER SYSTEM [14]

For the ratings such as the ones in Table 2, the utility function *u* can be defined in terms of ratings. The RS uses this function to compute the missing ratings to give recommendations.

Burke [7] divides RS's in four main classes in the way they create recommendations, namely: collaborative filtering (CF)-based, content-based, demographic and knowledge-based RS. There also exists 'Hybrid'-RS which are combinations of two or multiple of the classes above.

3.1.1 COLLABORATIVE-BASED

Collaborative RS's compute the utility of an item for a user *u*, based on the ratings (or other forms of transactions) from the other users *U'* in the dataset. The system first computes which users are most similar to *c* and based on the transactions of these users the utility is computed. Breese et al. [15] divide collaborative filtering into two classes: memory-based and model-based.

Memory-based algorithms use the complete set of ratings to compute recommendations, therefore the results are always up-to-date. Since it can only use common rated items, its performance decreases as the dataset gets sparser, i.e. relatively few recommendations compared to the number of users and products [16]. A rating can be computed as an aggregate or weighted sum over the ratings of similar users *U*'.

Sarwar et al [17] propose three approaches to compute the similarity between user *u* and *U'*, namely the Pearson correlation, cosine similarity or the adjusted cosine similarity. They also evaluated the performance of these similarity measures and found that the adjusted cosine resulted in the lowest Mean Absolute Error, see Figure 7: MAE performance of adjusted cosine, cosine and Pearson correlation. :

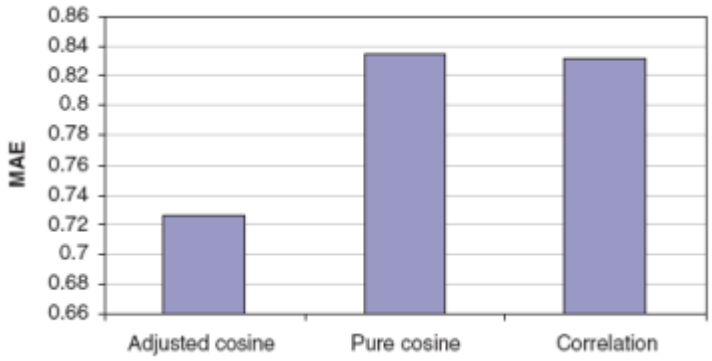


FIGURE 7: MAE PERFORMANCE OF ADJUSTED COSINE, COSINE AND PEARSON CORRELATION. [17]

3.1.1.1 CORRELATION-BASED SIMILARITY

Correlation-based similarity is often computed using the Pearson product-moment correlation coefficient (Pearson-R), but can also be computed using constrained Pearson correlation, Spearman rank correlation, and Kendall's correlation. [18] [16]

The Pearson-R correlation is based on the items that are rated by both users. Similarity scores are between -1 and 1. It has been used in for example TiVo, a show RS. [19]

$$sim(x,y) = \frac{\sum_{i=1}^{n}(x_i - \underline{x})(y_i - \underline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \underline{x})^2 \sum_{i=1}^{n}(y_i - \underline{y})^2}}$$

3.1.1.2 COSINE BASED

The ratings from users can be expressed as vectors. The cosine-based similarity computes the cosine of the angle between two rating vectors. We couldn't find any RS that uses this similarity function. Not every user uses the same scale for their ratings. The cosine-based similarity function does not take these user rating differences into account.

$$sim(x,y) = \frac{\sum_{i=1}^{n}(x_i)(y_i)}{\sqrt{\sum_{i=1}^{n}(x_i)^2 \sum_{i=1}^{n}(y_i)^2}}$$

3.1.1.3 ADJUSTED-COSINE BASED

The adjusted cosine is an improved cosine-based similarity function. It takes the differences in rating scales between users into account. Every rating is subtracted with the average rating of the user so the ratings are centered to this average. This function only takes ratings into account that are rated by both users.

$$sim(x,y) = \frac{\sum_{u \in U}(x_u - \overline{x_u})(y_u - \overline{y_u})}{\sqrt{\sum_{u \in U}(x_u - \overline{x_u})^2 \sum_{u \in U}(y_u - \overline{y_u})^2}}, where\ U = X \cap Y$$

Model-based collaborative filters create recommendations that are based on a model learned or estimated from the transactions [16]. Since it doesn't use the actual dataset it performs better if the dataset is sparse, because memory-based filtering only uses common items. The model is outdated if new information, like ratings, is available to the RS. In [20] various model-based approaches are mentioned, like Bayesian classifiers [21], neural networks [22], fuzzy systems [23], latent features [24] and matrix factorization [25].

3.1.2 CONTENT-BASED METHODS

Content-based RSs take a completely different approach for making recommendations. The utility function for new items is computed based on the similarity with other items the user has rated. So a RS for music videos could find similar items based on for example artists, songwriters, genres and similar lyrics.

Lops et al. [9] divide a content-based RS into three components, namely a

1. content-analyzer: a component that analyses new items and the features that can be used by the RS;
2. profile learner: a component that creates a user's profile. This can be done by explicit or implicit feedback from the user, and;
3. filtering component: the component that creates the actual recommendations.

The main advantage of content-based RSs is that they recommend only items for user *u*, based on the transactions of user *u* so transactions from other users are not needed. Another advantage is transparency: user *u* can get insight into why an item was recommended. They can then evaluate the importance of these features to determine the relevance of the recommendation.

Adomavicius et al. [14] describe several disadvantages of content-based RSs are mentioned, like limited content analysis, that can be improved using extensive domain knowledge to extract useful content that can be used to recommend new items. They also state that content-based techniques tend to recommend only similar items and only few unexpected items.

3.1.3 Demographic approaches

These RS's are based on a demographic profile of a user. This profile can be based on for example gender, age, owning a car and playing football. Recommendations are based on clusters that best match a user's profile. The process of this profiling is described in Figure 8.

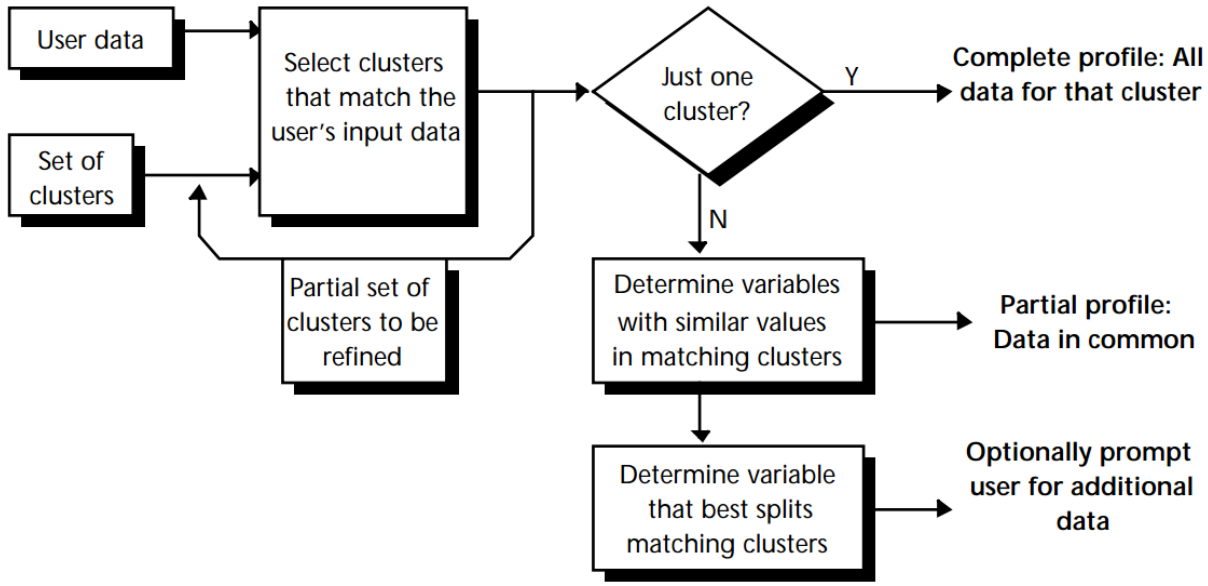


Figure 8: Basic process of demographic generalization. [26]

This approach is successfully evaluated by B. Krulwich [26] using a demographic system called PRIZM. He states that the demographic approach has the following advantages:

- self-learning;
- only minimal amount of information is needed from the user and
- it is possible to profile a user in areas not addressed by the input data if these new areas have relations with the input data.

3.1.4 KNOWLEDGE-BASED APPROACHES

RS's based on knowledge-based approaches recommends items based on the user's profile and the product information and does not depend on ratings. They try to interpret what products meet the user's requirements, via for example wizards.

Some RS's that are based on the knowledge-based approach are: FindMe [27], and Wasabi [28].

3.1.5 HYBRID APPROACHES

All RSs mentioned above have their advantages and disadvantages. To create better recommendations hybrid RS approaches can be built by:

- combining the predictions of collaborative and content-based RSs;
- use predictions of the best performing RSs dynamically or
- use features of content-based RSs in collaborative based filtering, or vice versa to create a new RS.

Therefore several papers have combined different methods to improve the performance of their RS, like Cavus et al. [29], Porcel et al. [30], Claypool et al. [31], Pazzani et al. [32] and Soboroff et al. [33].

3.1.6 PROBLEMS

Currently there are still several challenges to be resolved with RSs. In this section some of these problems are described.

3.1.6.1 COLD START

In most RSs the number of ratings is small compared to the ratings that need to be predicted, since not all items in a RS have been rated by all users. This makes it difficult to create reliable recommendations. Bobadilla et al. distinguished between three forms of cold start problems: [8] (1) new communities, (2) new items, and (3) new users. The new community problem is mostly related to new RSs, since the number of transactions in their system is small compared to the complete user-item space. The new item problem occurs for items that do not (yet) have many ratings. Especially if items have few high ratings they will not get recommended very often. This creates a vicious circle where these items remain unnoticed by users and therefore don't receive new ratings. The new item problem occurs mainly in CF-based RSs. The new user problem, finally, occurs for users that have added only a few transactions to the RS. The RS then cannot construct a good profile of this user.

3.1.6.2 SCALABILITY

As the number of users, items and/or transactions increases, the RS needs to process more data. This makes the calculation of new recommendations more complex, and therefore computationally intensive. Several attempts have been made to improve the scalability of RSs, like using Singular Value Decomposition (SVD) [34], distributed RSs [35] [36] [37], clustering [38] and incremental RSs [39].

3.1.6.3 CROSS-DOMAIN RECOMMENDATION

Current RSs are designed to create recommendations within a single domain, like music or books. Fernandez-Tobias presented several ideas to create cross-domain recommendations using Linked Open Data. [40]

3.1.6.4 CONSTRAINT-BASED RECOMMENDATIONS

Felfernig and Burke [41] describe a new sort of recommendation problem, namely constraint-based RS, where constraints could come from users or products. Constraint based recommendations become useful when recommendations should meet a set of requirements.

In this work, we focus on the cold start problem and the cross-domain recommendation problem

3.2 SEMANTIC WEB

Recently, Linked Data, a data publishing standard, was developed to enable a structured representation of online data [42]. This makes it possible to create typed links that are explicitly defined between documents from different sources. Since the links are explicitly defined, they can be read and understood by machines.

Links in documents are described in the Resource Description Framework (RDF) format [43]. RDF is a graph-based data model that makes it possible the make statements about resources. RDF encodes these statements using <subject, predicate, object> triples. The subject and object from such a triple are URIs that identify the resources. The predicate is also an URI that specifies the relationship between the subject and object. This is also known as the Web of Data.

Miller gives a graphical overview of RDF in [44]. The first example describes the data model using documents and authors. Figure 9 describes “Document 1” that is written by an author with a name, email and affiliation.

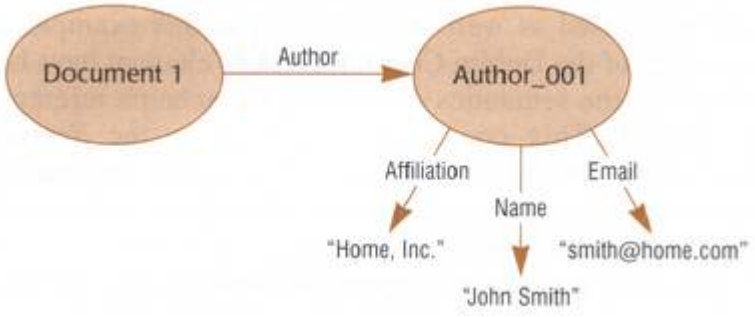


FIGURE 9: RDF DATA MODEL DESCRIBING A DOCUMENT AND THE AUTHOR. [44]

Linked Data is built on two technologies that are already used by the World Wide Web: Uniform Resource Describers (URIs) and the HyperText Transfer Protocol (HTTP). URIs are based on Uniform Resource Locators (URLs) that are used to locate documents on the Web. URIs are used to locate and identify entities, instead of web documents. Entities can be looked up using the URI over for example the HTTP protocol.

The data model presented in Figure 9 can be easily understood by humans. However, a more detailed syntax is required to store this model in machine readable files. The property “name” can be used for different objects, meaning different things. This could lead to inconsistent representations of the semantics. RDF uses XML to support consistent representations of the semantics.

XML namespaces make it possible to unambiguously identify the semantics of property types. Therefore, RDF tuples are described using vocabularies that contain collections of classes and properties. These vocabularies are based on the RDF Vocabulary Definition Language (RDFS) or the Web Ontology Language (OWL). An example of such a vocabulary is the Dublin Core Initiative. They define an "author" as the "person or organization responsible for the creation of the intellectual content of the resource" in the Dublin Core CREATOR element (DCES). The data model from Figure 6 is rewritten using the Dublin Core vocabulary in Figure 10.

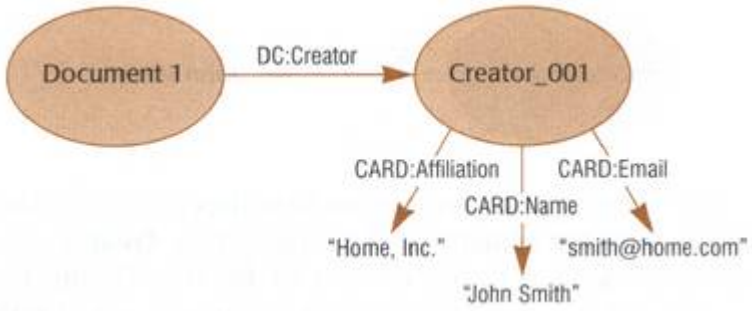


FIGURE 10: DATA MODEL DESCRIBED USING THE DUBLIN CORE VOCABULARY. [44]

The Linked Data principles are used in the Linked Open Data (LOD) project. In Figure 11 an overview is given of the published data sets that are included in the LOD in 2014.

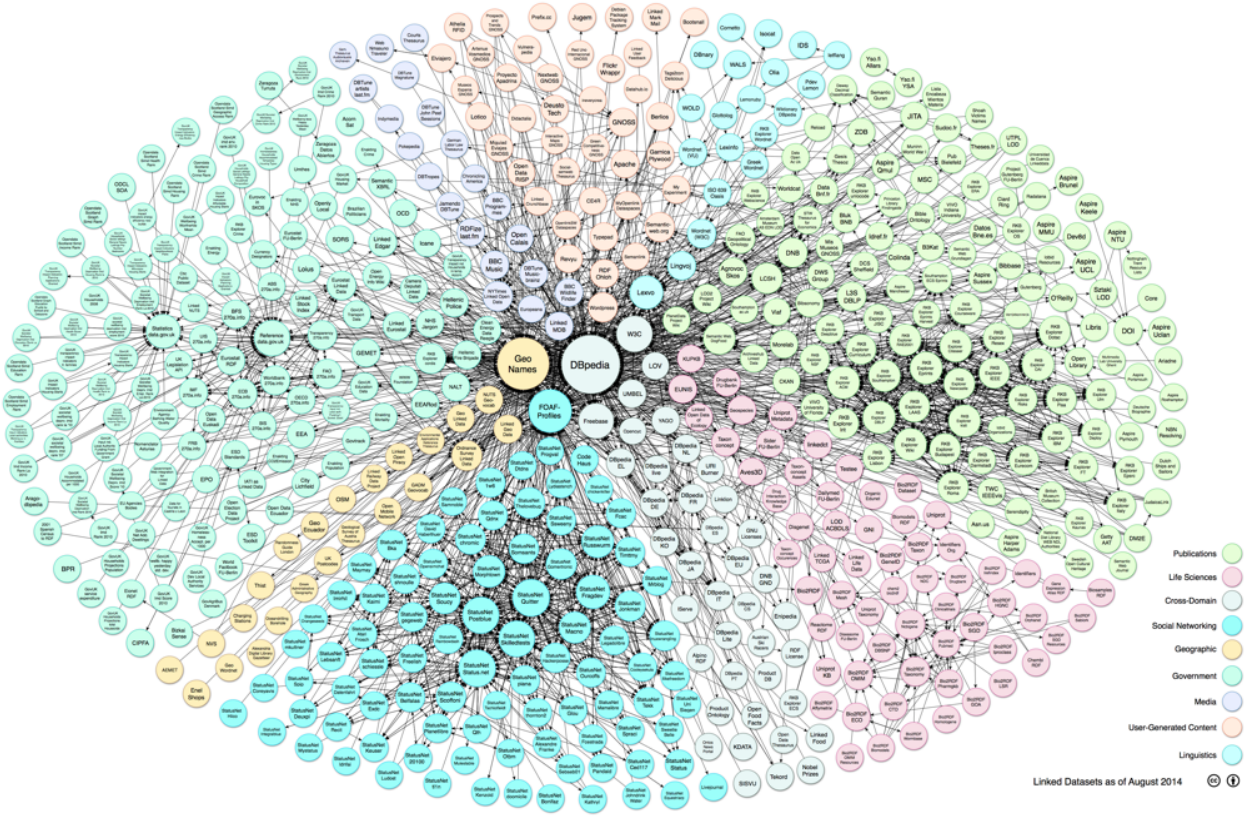


FIGURE 11: LINKED OPEN DATA (LOD) CLOUD ON AUGUST 30TH, 2014. [45]

Looking at Figure 11, the central node DBpedia is one of the largest published data sets. The DBpedia project extracts the structured data from the Wikipedia dataset [46]. DBpedia is *"a crowd-sourced community effort to extract structured information from Wikipedia and make this information available on the Web"* [47]. This structured information can be found for example in the infoboxes on the right-hand side and the categorization information on the bottom of Wikipedia pages. DBpedia uses RDF to represent the extracted information about the entities. The DBpedia set contains a total of 3 billion RDF triples, including 580 million extracted from the English edition alone.

3.2.1 RELEVANCE SEARCH

The information in RDF graphs can be used to find similar or related items. Items are represented by nodes in a RDF graphs. Since nodes are connected in a graph it is possible to find paths between nodes.

Several approaches to find relevant paths between nodes in RDF graphs have been proposed. Some focus on finding related items in homogeneous networks, i.e. networks where each entity has the same type while other focus on heterogeneous networks, where entities have different types. Items can be connected via different paths, where each path can have a different meaning.

Chakrabarti introduces SimRank [48], a similarity measure that can be used on homogeneous networks. It is based on the intuition that "two objects are similar if they are referenced by similar objects", with the base case that an object is maximally similar to itself.

In a heterogeneous network can be explained using two concepts: Network schema and meta-paths [49]. A network schema describes all possible entities in a network, including their relations. A meta-path describes how two entities are related using the network schema. Chuan Shi et al [50] use the conference-domain to describe heterogeneous networks, as seen in Figure 5.

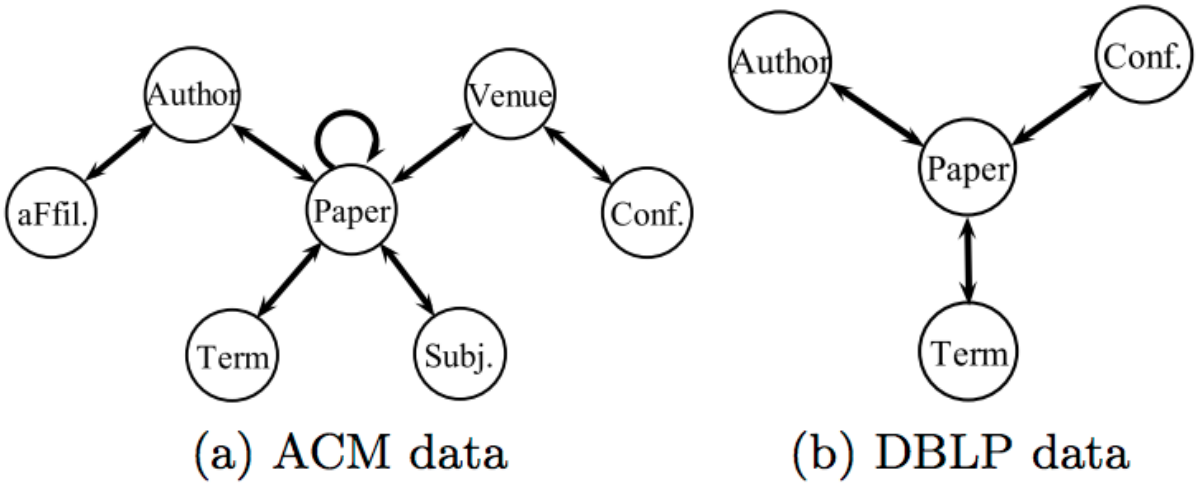


FIGURE 12: HETEROGENEOUS INFORMATION NETWORKS. [50]

In Figure 12, two information networks are presented from based on different data models. In Figure 12 (a) conferences and authors can be connected using different metapaths.

1. Author-Paper-Venue-Conference (APVC), meaning an author has written a paper that is presented on a conference
2. Author-Paper-Subject-Paper-Venue-Conference (APSPVC), meaning an author has written a paper on a subject, and another paper with the same subject is presented on a conference.

These meta paths can be used to find similar items. Similarity measures that have been applied in information network mining include [51]:

1. path count, the number of existing paths between two items;
2. random walk, the probability that a random walk starting from one item, ends in the other item and
3. pairwise random walk, the probability that two random walks, starting from two different nodes end up in the same item in the middle.

A drawback of these similarity measures it the relatively high visibility of nodes that are connected to a high number of edges. Path count and random walks are likely to end up in more concentrated nodes. However, in many applications similar items should also have the same visibility.

There exist algorithms that also incorporate the visibility into their relevance measure. Sun et al have proposed an algorithm called PathSim [51] for same typed nodes based symmetric paths. They incorporate visibility by dividing the total number of paths instances between two nodes by the number path instances between themselves.

Another similarity measure that works for heterogeneous nodes in a heterogeneous network is HeteSim [50]. It is based on a pair wise random walk and has some useful properties like having a self-maximum and being symmetric.

3.3 EXISTING APPLICATIONS

In this section we provide an overview of Linked Data-based RSs.

3.3.1 MOVIEXPLAIN: A RECOMMENDER SYSTEM WITH EXPLANATIONS

Symeonidis et al present the RS MoviExplain [52]. This is a RS for movies that is based on the following features: genres, directors and actors. The rating for a movie is used as a rating for the features belonging to that movie. The RS uses these features to construct an ordered list of new movies that match the features of rated movies. This list also contains explanations why movies are present, based on the features of the movies. According to Symeonides, users appreciated this explanation type, because they could easily judge recommendation relevance.

3.3.2 LINKED OPEN DATA FOR CONTENT-BASED RECOMMENDER SYSTEMS

Mirizzi et al. presented More than Movie Recommendation (MORE), a movie recommender system that creates movie recommendations based on two RDF datasets, DBpedia and LinkedMDB. Recommendations are based on similarities between two movies. The similarities can be computed if they are:

- directly related;
- the subject of two RDF triples having the same property and the same object, as for example when two movies have the same director, or;
- the object of two RDF triples having the same property and the same subject.

The movies are represented in 3-dimensional tensor where each slice refers to an ontology property, as seen Figure 13.

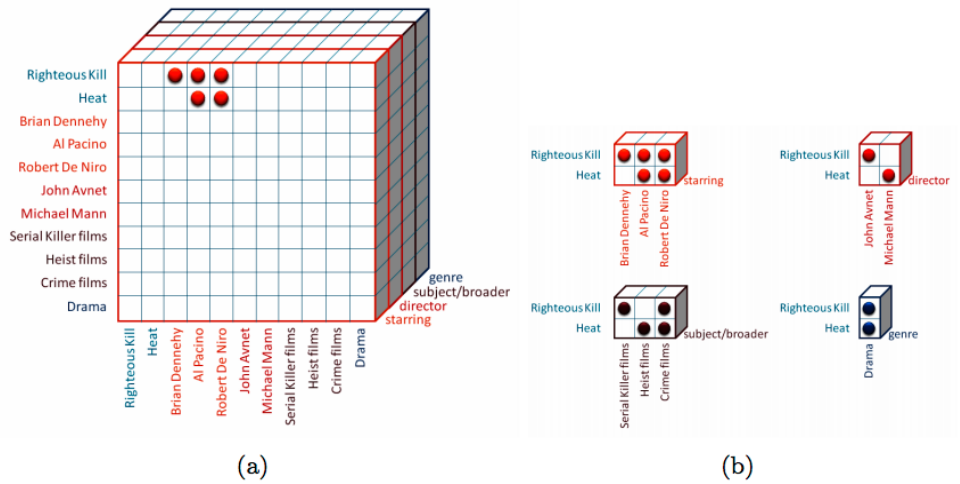


FIGURE 13: (A) TENSOR REPRESENTATION OF THE RDF GRAPH. (B) SLICES DECOMPOSITION. [53]

Given a property, each movie is seen as a vector, whose components refer to the term frequency-inverse document frequency (TF-IDF). The similarity between two movies is the cosine of the angle between the vectors, representing those movies.

3.3.3 HETERECOM: A SEMANTIC-BASED RECOMMENDATION SYSTEM IN HETEROGENEOUS NETWORKS

Chuan Shi et al. present HeteRecom [54] a RS is presented that is based on the HeteSim [55] algorithm. First all the relevance scores between items in the Heterogeneous Information Network (HIN) are computed offline and stored in a matrix. Different paths between same objects are weighted based on the number of average in- and outlinks.

3.3.4 DBREC - MUSIC RECOMMENDATIONS USING DBPEDIA

Passant presents dbrec [56], a RS that gives recommendations based on RDF data from DBpedia. The list of recommendations is constructed based on a list of related artists. To compute the relatedness between artists, the writers defined the Linked Data Semantic Distance (LDSD) measure. In this paper this measure only considers directly linked resources or through a third resource. Recursive patterns (like SimRank) may be used in the future. Via experiments they identified a list of useful properties in the DBpedia dataset. Evaluations show that the average mark for the recommendations was 3.4 (LastFM was 3.69) and also the precision was quite good, 92, where Last.FM had 98.3). Users also liked the explanations of the presented recommendations, as seen in Figure 14.

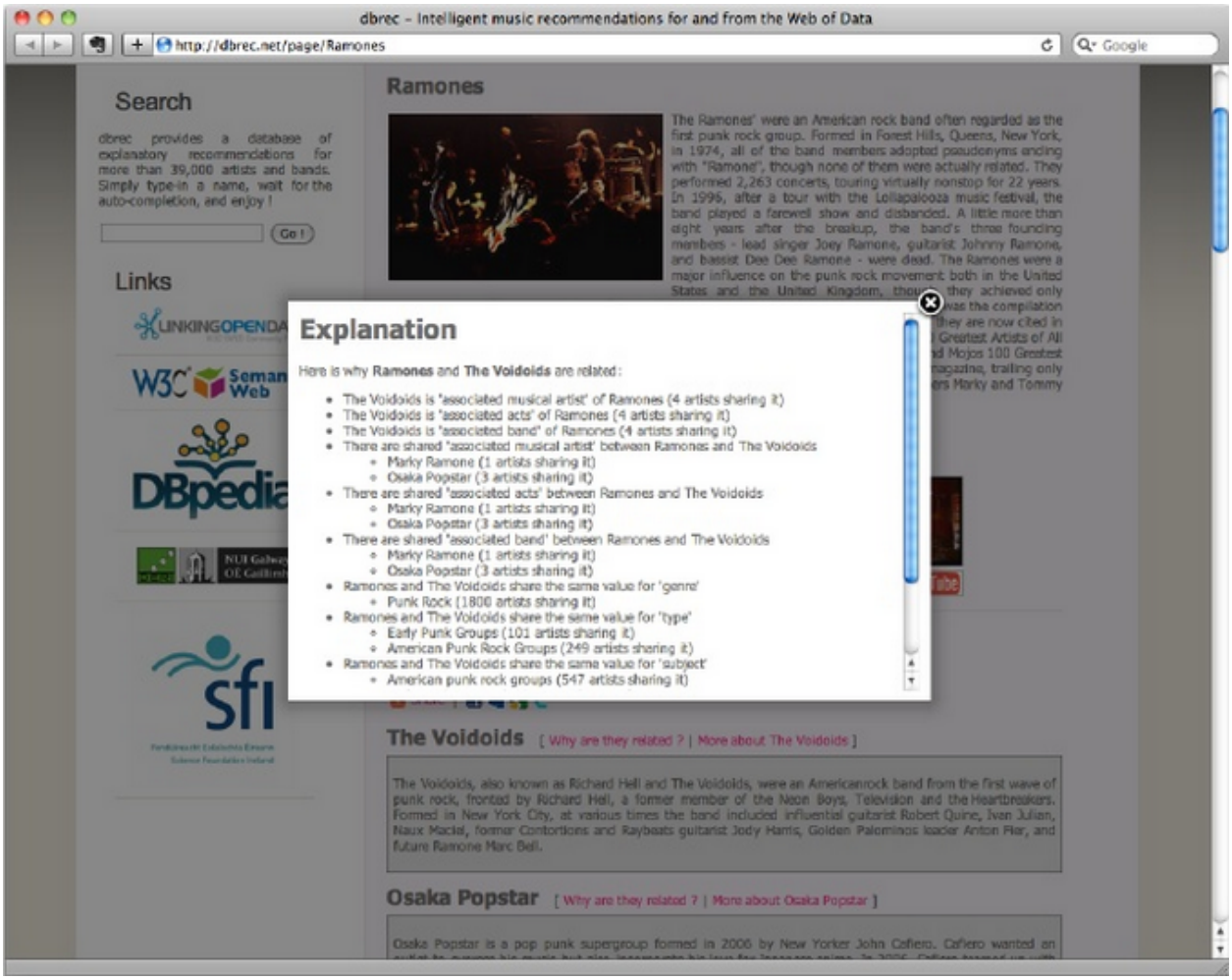


FIGURE 14: INTERFACE OF DBREC. [56]

3.3.5 SPRANK

Ostuni et al. presented the "Semantic Path-based Ranking (SPRank) [57]. This is a RS that first explorer paths in a semantic graph to find related items. From these paths several path-based features are extracted and are inserted into a learning to rank algorithm the recommend the most relevant items.

3.4 CONCLUSION

In this thesis we introduce IBRS that uses structured information from LOD to find related items. Using LOD can improve several shortcomings of existing RSs. With IBRS, we focus on the cold-start problem and recommendations in multiple domains.

The cold-start problem means that the RS has insufficient data to create a good profile for a user. Our aim is to improve this by using profiles from other websites. Since Facebook has a good API for fetching a user's profile and it have a lot of user, it is a good place to start creating profiles for our RS. IBRS collects all the Facebook likes from new users. These likes are mapped to resources in the DBpedia dataset.

IBRS should be able to create domain-independent and language-independent recommendations. Semantic relations in the LOD are used to find related content. The tags from the tagged item set in IBRS are used to extract tags from the related LOD-items. Now our RS can act like a content-based RS to extract. To measure the importance of the tags TF-IDF could be used.

CHAPTER 4:

TECHNOLOGY

This chapter describes how the RDF graph from DBpedia can be used to implement IBRS. In this version of IBRS we implemented the dataset from Kaartje2go as explained in Chapter 2 to recommend postcards.

In Section 4.1 a global overview of IBRS is given. In Section 4.2 we describe how we implemented IBRS for online recommendations. The individual components are explained in more detail in Section 4.3.

4.1 GLOBAL OVERVIEW

IBRS is based on the idea that people might be interested in items that are related to things they already like. Social media websites are great platforms for expressing these interests to other users, or IBRS.

The interests that people publish on social media websites are often in a different domain than the item set in IBRS. However, these interests may contain pointers to items that are available in IBRS that can be recommended. For example a user that is interested in 'Cristiano Ronaldo' is probably also interested in 'football' and the football club 'Real Madrid'.

The first step for creating recommendations is getting a list of interested items from the user. This set of interesting items is obtained from his social media profile, but could just as well be a result of free-text user input. In this thesis we use the likes from Facebook as input for IBRS.

To find related items within the DBpedia RDF graph *G*, the user's interests will be matched to items in *G*, using the *Facebook-Mapper*. The next step is to explore *G* and find related items in DBpedia via semantic paths, using the *DBpedia Explorer*. This step results in a relatively small subgraph of *G*, that we call *G'*. This subgraph only contains the relations between the user's input and the tags. It is important to keep *G'* as small as possible, since we are only interested in useful related items.

To create recommendations IBRS transforms the set of DBpedia resources to tags using the "*Recommendations Creator*" component. This component returns a subset of the tags that are already present in the IBRS data set and where the user is probably interested in. These tags are used to create the recommendations. Because IBRS does not contain any knowledge itself, but combines information from different sources, IBRS can recommend any tagged object set, independent of language and domain.

The global overview is depicted in Figure 15.

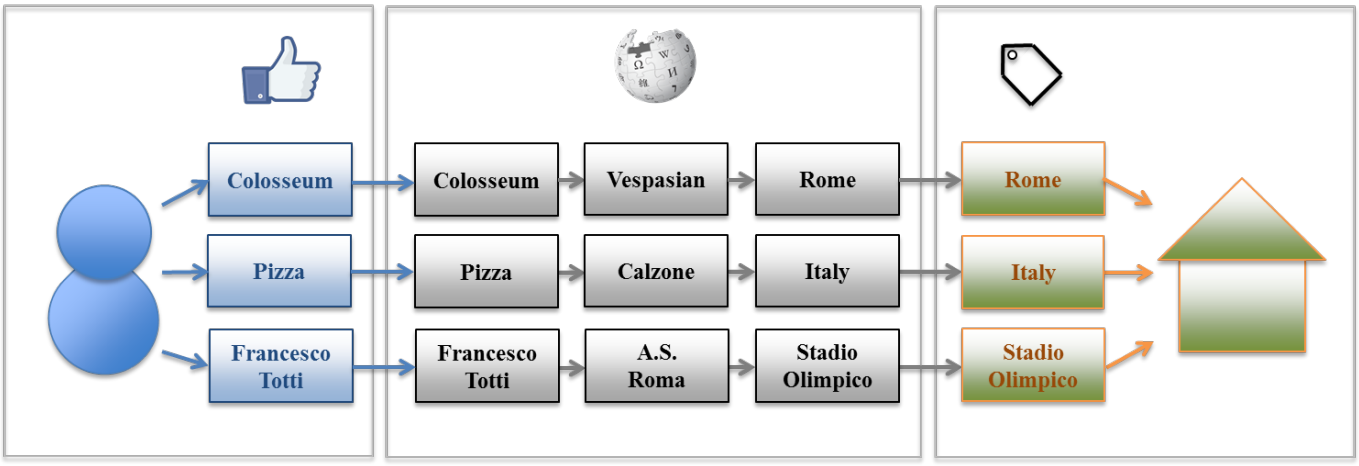


FIGURE 15: GLOBAL OVERVIEW OF IBRS.

To clarify this overview, several examples are given in Figure 16. In these examples the Facebook page about 'Cristiano Ronaldo' is taken as input for the system. In the first example only the DBpedia resource about Cristiano Ronaldo is used to extract the tags *'Madrid'*, *'Portugal'*, *'Lissabon'* and *'Football'*. The second example uses '*Real Madrid C.F.*', which is connected to '*Cristiano Ronaldo*' via the 'team'-property, coming from the ontology *http://dbpedia.org/ontology/team*. This relation results in the tags *'Madrid'*, *'Spain'*, *'Soccer'*. It is possible that these properties create a cycle, as seen in Figure 16(d).

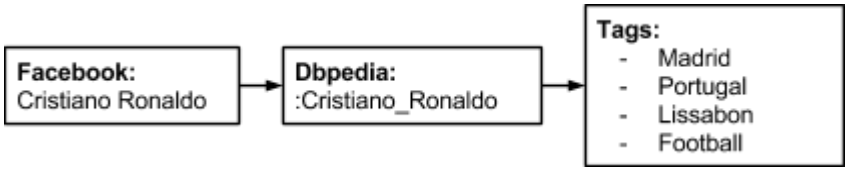


FIGURE 16(A): FACEBOOK PAGE ABOUT CRISTIANO RONALDO LEADS TO THE TAGS MADRID, PORTUGAL, LISSABON AND FOOTBALL.

FIGURE 16(A): FACEBOOK PAGE ABOUT CRISTIANO RONALDO IS CONNECTED TO REAL MADRID C.F. AND LEADS TO THE TAGS MADRID, SPAIN AND SOCCER.

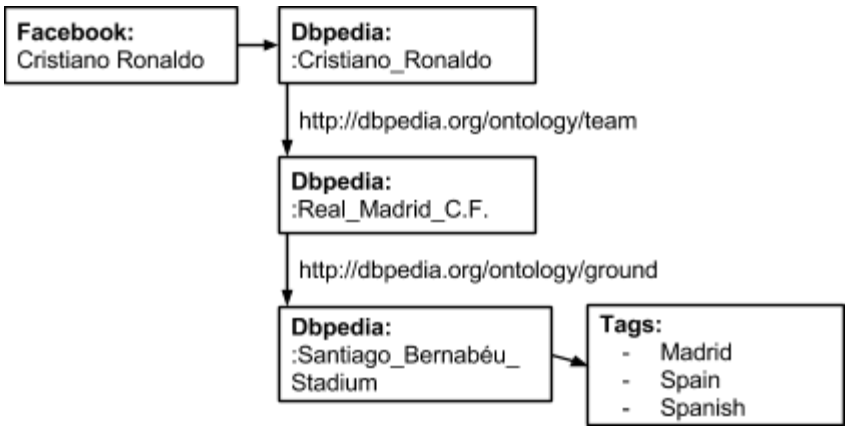


FIGURE 16(C): FACEBOOK PAGE ABOUT CRISTIANO RONALDO IS MAPPED TO THE DBPEDIA RESOURCE WHICH IS CONNECTED TO SANTIAGO BERNABEU VIA REAL MADRID C.F. THIS RESULTS IN THE TAGS MADRID, SPAIN AND SPANISH.

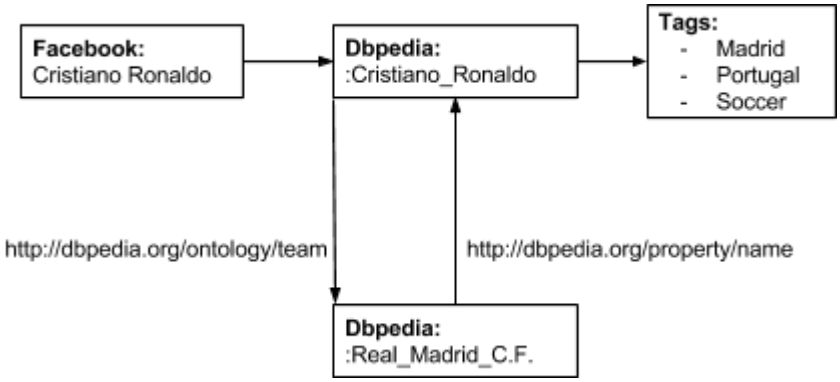


FIGURE 16(D): CRISTIANO RONALDO CAN BE REACHED VIA REAL MADRID C.F. AND RESULTS IN THE TAGS SOCCER AND PORTUGAL.

4.2 IBRS

To demonstrate the interest-based RS, we build the Interest-Based Recommender System (IBRS).

This RS is built as a web application written in PHP. The following libraries were used to develop the system:

- CakePHP 2.6.3, an open-source web application framework written in PHP;
- EasyRDF 0.9.0, a PHP framework for producing and consuming RDF;
- Facebook PHP SDK 4.0.20. the official SDK to connect to the Facebook API, and;
- Bootstrap 3.3.2, a HTML, CSS, and JS framework for the graphical user interface (GUI).

All the data is stored in MySQL database. To create a generic RS that can be used with different item sets, an abstraction layer is used on top of the product table(s). In this thesis, we used IBRS with two product sets, namely cards and holiday homes (properties), as seen in Figure 17.

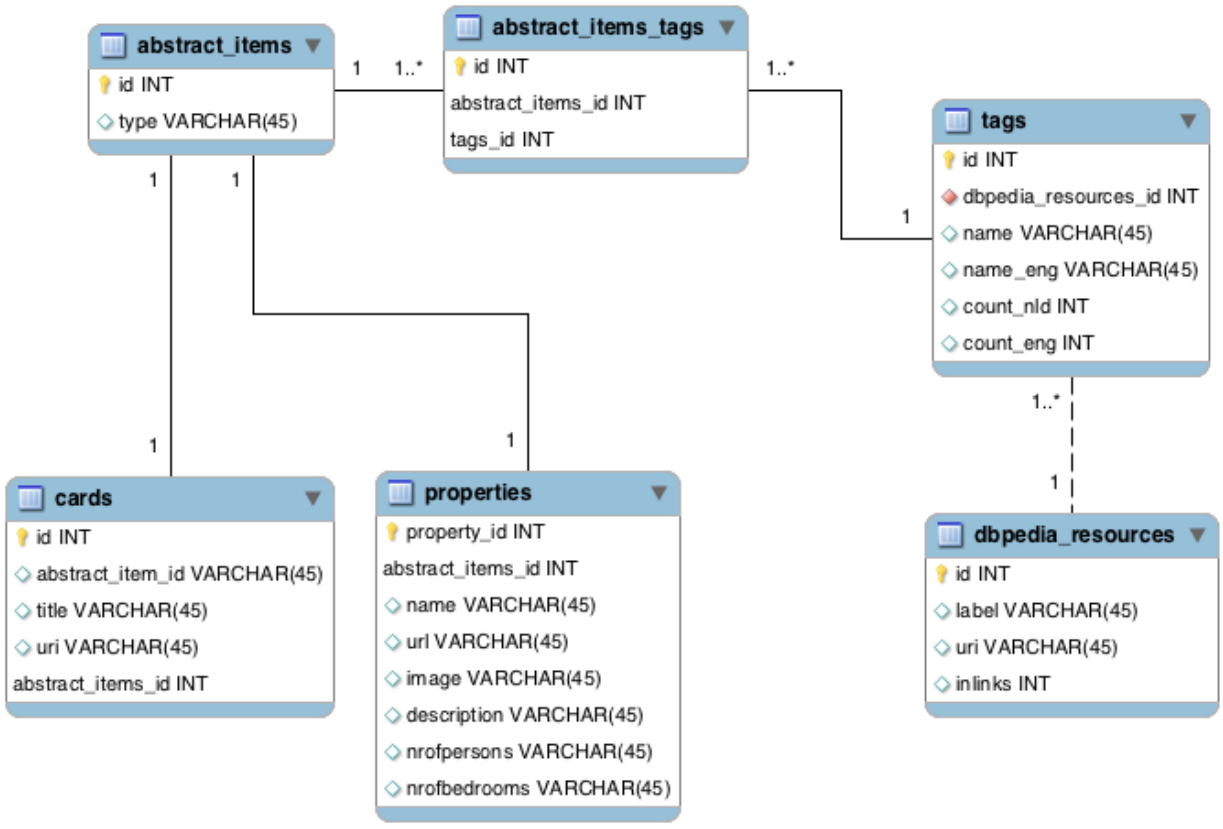


FIGURE 17: SCHEMATIC OVERVIEW OF THE DATABASE USED IN IBRS.

The following tables are used in IBRS:

- *abstract_items*: all items that can be recommended are stored in this table. This table is used as a base class for either *cards* or *properties*;
- *abstract_items_tags*: every row from the table *abstract_items* has many tags. These tags are stored in the table *tags*, via the join-table *abstract_items_tags*;
- *tags*: this table contains the keywords that are associated with the items in either the 'cards' or 'properties' table. Every tag has a Dutch and English name and the corresponding word count in all Wikipedia abstracts and
- *dbpedia_resources:* A dbpedia_resource contains metadata about the DBpedia resource, like the URI and the number of inlinks.

Since the algorithm may take a while to complete, some server parameters need to be changed. Some important server configurations settings can be found in Table 3.

Setting	Value
Apache Version	Apache/2.2.22 (Ubuntu)
memory_limit	1024M
default_socket_timeout	60
MySQL version	5.5.29

TABLE 3: SERVER CONFIGURATION SETTINGS

4.3 DETAILED OVERVIEW

In the following subsections the several individual components of ARS are described: the *Facebook mapper* for mapping Facebook-likes onto the DBpedia resources, the *DBpedia explorer* to find related resources, the *Tag Selector* for selecting useful tags from the tagged item set and finally *the Recommendations Creator* for generating the recommended items.

4.3.1 FACEBOOK MAPPER

The input of the RS is a set of liked pages from a user on Facebook. These liked pages define the starting nodes in the RDF graph from DBpedia. In the first step we need to map the Facebook likes of a user to DBpedia resources.

To collect the Facebook likes we use the Facebook Graph API 2.3. The collection of liked pages is a set of 'Page'-nodes, where each Page is defined by the following fields:

- Facebook id
- name
- category

Using this information, we need to query the DBpedia SPARQL endpoint in order to find out if a Facebook like can be mapped to a DBpedia resource.

The simplest way is to check if a DBpedia resource exists with identical the same name as a Facebook like. Unfortunately, this would result in a lot of mismatches, due to ambiguity. Some examples of ambiguity we found include:

- Up: The movie "Up" is a movie that is published in 2004. However, a movie with the same name was also released in 1984. The name "Up" is also used in various other things in several domains. It was the name of a Volkswagen car, a song by R.E.M. and a shorthand name of the University of Phoenix.
- Apple: An apple is of course a fruit that grows in a tree. But is also known as an alias for the technology company "Apple Inc.". Luckily, the most popular Facebook page about "Apple Inc." has the correct name. Since probably not all iPod, iPad, iPhones or MacBook's will like apples this may result in incorrect recommendations.

To overcome this problem, we use append some category names in the sparql query. A complete list of used categories is presented in Table 4.

Facebook category	Appended after search term
"TV show"	"_(TV_series)"
"Movie"	"_(movie)"
"Musician/band"	"_(band)"

TABLE 4: META INFORMATION THAT IS ADDED TO THE URI'S

Using this meta data improved the mapping results with 13% based on a test set of 11674 Facebook pages. Without the meta information 218 pages were not found and 41 wrong pages were found.

The complete sparql query can be found in Table 5. This queries uses the following three variables:

- $metaUri: The complete DBpedia URI with meta description, for example: http://dbpedia.org/resource/Game_of_Thrones_(TV_series)
- $uri: The complete DBpedia URI without meta description, for example: http://dbpedia.org/resource/Game_of_Thrones
- $tag: The name of resource, for example "Game of Thrones"

Sparql	**Explanation**
PREFIX dbpont: <http://dbpedia.org/ontology/>	Add prefix for the DBpedia "ontology"
SELECT ?uri ?label ?abstract	Select uri's, labels and abstracts
WHERE {	
{ ?uri dbpont:wikiPageID []. FILTER(?uri = <{$metaUri}>) }	uri has exact match with $metaUri and is connected to a Wikipage
UNION { ?uri dbpont:wikiPageID []. FILTER(?uri = <{$uri}>) }	OR uri has exact match with $uri and is connected to a Wikipage
UNION { <{$metaUri}> dbpont:wikiPageRedirects ?uri}	OR DBpedia resource can be found using a redirect from $metaUri
UNION { <{$uri}> dbpont:wikiPageRedirects ?uri}	OR DBpedia resource can be found using a redirect from $uri
UNION {?uri rdfs:label \"{$tag}\"@nl.}	OR DBpedia has a label equal to $tag
?uri rdfs:label ?label.	DBpedia resource should have a label
?uri dbpont:wikiPageID ?wikiPageid.	DBpedia resource should be connected to a Wikipage
?uri ?p1 ?o2 .	DBpedia resource should be connected to a different DBpedia resource
?uri dbpont:abstract ?abstract .	DBpedia resource should have an abstract
FILTER (LANG(?abstract) = \"nl\") .	Only retrieve Dutch abstracts
FILTER (langMatches(lang(?label),\"nl\")).	Only retrieve Dutch labels
MINUS {?uri rdf:type skos:Concept}	Filter DBpedia Concepts, like 'Categories'
}	
LIMIT 1	Only return a single DBpedia resource

TABLE 5: USED SPARQL QUERY

To evaluate the performance of the usefulness of Facebook likes, we collected the Facebook likes of 309 users and checked how many likes can be used in the RS. These 309 users have liked 18,078 (11,674 unique) Facebook pages together, which is on average 58.5 likes per Facebook user.

Most of these Facebook users are Dutch, so the unique likes are mapped to the English and Dutch version of DBpedia. These results are presented in Figure 18: Facebook likes map results for the Dutch and English DBpedia dataset..

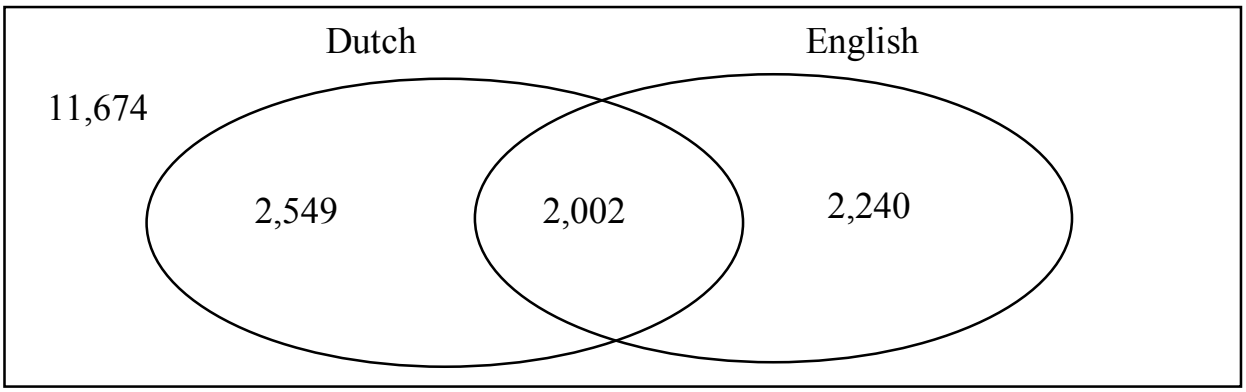


FIGURE 18: FACEBOOK LIKES MAP RESULTS FOR THE DUTCH AND ENGLISH DBPEDIA DATASET.

So from the 11,674 unique Facebook likes, 2,549 likes (21.83%) could be mapped to the Dutch version of the DBpedia set and 2,240 likes (19.19%) could be mapped to the English version. Even though these percentages may be low at first sight, a user can still be profiled using almost 15 likes on average.

4.3.2 DBPEDIA EXPLORER

We now have a set of Facebook likes that are mapped to the DBpedia dataset. This gives a set of initial nodes in the DBpedia LOD. The next step is to traverse this LOD to find related items that can be used to create recommendations.

As described in Chapter 2, nodes in LOD are connected via directed predicate. So to find related items you can simply follow the relations from one node to their neighbors.

To improve the precision of the RS it is important to make a good selection of related DBpedia resources. If the DBpedia explorer find too many related DBpedia resources it has a negative influence on the precision and speed, since more resources should be analyzed. On the other hand, it could have a beneficial impact on the recall since more resources are used by the RS.

Mirizzi et al. [58] found that the best path length for finding related items was 2. This means that related DBpedia resources can be reached via a single intermediate node. There are four different ways of creating links using a path length of 2, as indicated in Figure 1. It is possible that the starting node A and final node C are the same node.

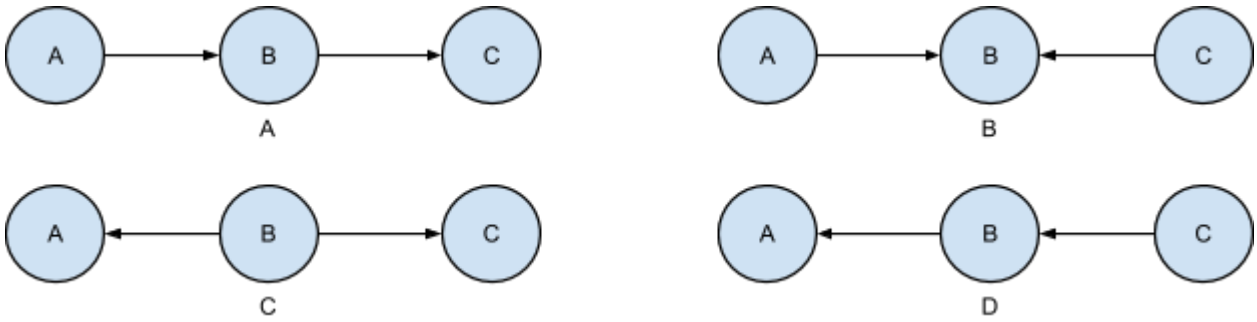


FIGURE 19: FOUR DIFFERENT WAYS OF SELECTING RELATED ITEMS C VIA A MIDDLE NODE B, WHERE RELATED ITEMS ARE FOUND USING (A) ONLY OUTLINKS, (B) OUTLINKS FROM NODE A AND INLINKS FROM NODE B, (C) INLINKS FROM A AND OUTLINKS FROM B AND (D) ONLY USING INLINKS.

Traversing different paths results in different sets of related items, as each path has its own semantic meaning. The path A → B could be a direct relation, as seen in Figure 20(a) but it also results in moving towards broader concepts, like Figure 20(b).

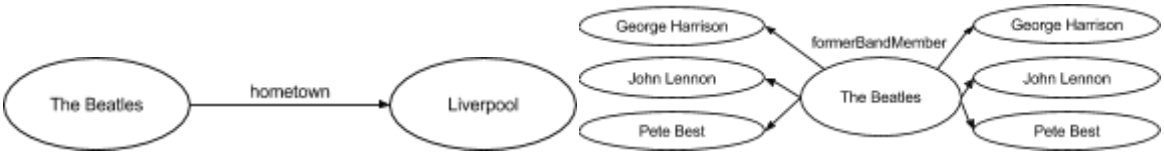


FIGURE 20: (A) DIRECT RELATION FROM THE BEATLES (B) THE BEATLES MOVING TOWARDS BROADER CONCEPTS.

To compare these different result sets, we run the algorithm from three different starting node, namely:

- Cristiano Ronaldo, a Portuguese football player;
- Amsterdam, the capital of the Netherlands and
- Arctic Monkeys, an indie rock band from England

The SPARQL endpoint for the English version of DBpedia (http://dbpedia.org/sparql) is used to get the result sets. We compare the following metrics for the different paths:

- size, since some paths will result in more related items
- speed, if some paths result in bigger result sets, querying and fetching the results will probably take more time
- top 10, some related items can be reached via multiple middle nodes. This indicates the most related C's for a starting node A.

In Table 6the results for Cristiano Ronaldo <http://dbpedia.org/resource/Cristiano_Ronaldo> are given. The first path contains the least, but most useful related pages to Cristiano Ronaldo, like football leagues and

clubs he played for and his birthplace. The other paths result mostly in other football players. The second path leads to the largest set, because there are many football player (C's) that have played at football teams (B's) where Cristiano Ronaldo (A) have played.

	A → B → C	**A → B ← C**	**A ← B → C**	**A ← B ← C**
Size	344	666760	3233	1898
Speed (s)	0.0842	0.0773	0.0812	0.0866
Top 10	Primeira Liga (15) S.L. Benfica (10) C. Ronaldo (10) La Liga (9) F.C. Porto (8) S.C. Braga (8) SC. de Portugal (8) Captain (association football) (8) J.M. Eduardo (7) Funchal (7)	C. Ronaldo (94) H. Porfírio (43) P. Alves (40) D. da Cruz Carvalho (40) Nani (38) R. Carvalho (38) H. Viana (37) H. Postiga (36) L. Figo (36) J.A. Lima (36)	C. Ronaldo (147) Real Madrid C.F. (76) Santiago Bernabéu Stadium (61) F. Pérez (52) X. Alonso (43) S. Ramos (42) Pepe (footballer born 1983) (40) I. Casillas (39) M. Vieira (39) Á. di María (34)	P. Bento (15) A.. Oliveira (15) J.A. Costa (15) H. Coelho (15) J.A. de Almeida (15) 2010–11 Real Madrid C.F. season (14) 2011–12 Real Madrid C.F. season (13) 2012–13 Real Madrid C.F. season (13) J.M. Pedroto (12) F. Peyroteo (12)

TABLE 6: RELATED ITEMS RESULTS FOR CRISTIANO RONALDO.

In Table 7 an overview is given of the found related items for Amsterdam <http://dbpedia.org/resource/Amsterdam>

	A → B → C	**A → B ← C**	**A ← B → C**	**A ← B ← C**
Size	117	243148	10292	23599
Speed (s)	0.081	0.934	0.409	0.169
Top 10	Netherlands (67) Central European Time (66) Amsterdam (52) North Holland (52) List of municipalities of the Netherlands (24) Municipal council (Netherlands) (24) List of sovereign states (17) Provinces of the Netherlands (16) Postal codes in the Netherlands (12) Telephone numbers in the Netherlands (12)	Amsterdam (112) Hilversum (39) Rotterdam (39) Haarlem (38) Oostzaan (37) Naarden (36) Weesp (36) Heemstede (36) Zaanstad (36) Enkhuizen (35)	Amsterdam (11882) Netherlands (9610) AFC Ajax (1062) North Holland (528) Midfielder (513) Painting (410) Central European Time (400) Dutch Republic (396) Netherlands national football team (367) Defender (association football) (360)	List of Dutch football transfers summer 2010–11 (172) List of Dutch football transfers summer 2009 (104) List of Dutch football transfers summer 2008 (98) List of Dutch football transfers winter 2009–10 (62) Amsterdam (52) List of Dutch football transfers winter 2010–11 (52) C. de Graeff (42) J. den Uyl (37) E. van Thijn (36) J.R. Thorbecke (34)

TABLE 7: FOUND RELATED ITEMS FOR AMSTERDAM.

The first path leads to very central nodes like "Central European Time" and "* in the Netherlands". The other paths also lead to nodes that not seem to be very related to Amsterdam. Again, the first path leads to the smallest number of related pages and is therefore the quickest.

Table 8 gives an overview of related items for the British indie-rock band Arctic Monkeys.

	A → B → C	**A → B ← C**	**A ← B → C**	**A ← B ← C**
Size	223	22766	2936	568
Speed (s)	0.083	0.147	0.104	0.083
Top 10	Indie rock (46) Arctic Monkeys (41) Garage rock (38) Sheffield (33) England (28) Post-punk revival (24) Bass guitar (22) Psychedelic rock (22) Domino Recording Company (22) Reverend and The Makers (21)	Arctic Monkeys (72) The Death Ramps (40) Miles Kane (35) Matt Helders (32) Alex Turner (musician) (31) Teddy Picker (28) One for the Road (Arctic Monkeys song) (28) Alison Mosshart (26) Don't Sit Down 'Cause I've Moved Your Chair (26) Andy Nicholson (23)	Arctic Monkeys (605) Indie rock (292) Alex Turner (musician) (234) Domino Recording Company (233) Psychedelic rock (166) Garage rock (138) Post-punk revival (124) James Ford (musician) (120) Gramophone record (88) England (81)	Arctic Monkeys (41) The Death Ramps (34) The Rascals (English band) (33) Alex Turner (musician) (33) Fluorescent Adolescent (33) One for the Road (Arctic Monkeys song) (32) Humbug (album) (30) The Bottletop Band (30) When the Sun Goes Down (Arctic Monkeys song) (30) The Hellcat Spangled Shalalala (30)

TABLE 8: RELATED ITEMS FOR ARCTIC MONKEYS.

The last second path results again in the least interesting nodes, since it mostly contains albums and songs from the Arctic Monkeys. The third path leads to the best results, since it contains the genre, label, related artist and the country. Via the first path also similar related items are found, but not as many as the using the third relation, A ← B → C.

Because tag sets typically contain broad concepts rather than instances thereof, the first path is the most useful for our RS. The tag set from our case study at Kaartje2go for example contains 138 cards with the tag “football”, but none with Cristiano Ronaldo.

4.3.3 TAG SELECTOR

The next step in the recommendation process is generating related tags. After the DBpedia explorer have found the related DBpedia items, the related items needs to be analyzed and converted to a set of tags. In this Section we use the tags in the dataset from Kaartje2go.

Table 1 gives an overview of the top-20 most used tags in this dataset. Most of these tags are not useful for recommending new items, since they

1. not capture the content of the postcard and
2. will not be found in DBpedia dataset.

Since we are using the DBpedia dataset to find tags, we can remove tags from our dataset that cannot be found in the DBpedia dataset. The Facebook mapper (Section 4.1) was used to find if an entry exists in the DBpedia dataset for a tag.

After cleaning the dataset, the total number of mapped tags was reduced from 46,337 to 38,388 the most used tags can be found in Table 9.

Tag	# postcards	Tag	# postcards
birthday	7,534	condolence	1,264
party	3,761	wood	1,182
girl	2,221	jubilee	1,149
love	2,136	wedding	1,092
baby	2,056	friendship	1,090
boy	2,039	illustration	1,026
photo	1,793	new year	999
marriage	1,535	mourning	861
tough	1,381	in love	821
animals	1,373	happy birthday	746

TABLE 9: CLEANED TOP 20 TAGS.

We now have a set of DBpedia resources (with metadata) and a set of tags. To extract a ranked list of useful tags the abstracts are used of the Wiki pages that belong to the related DBpedia resources. The algorithm works as follows:

1. All abstracts are collected from the related DBpedia resources.
2. These abstracts are concatenated using the ‘ ‘ (space) character.
3. For every tag the number of occurrences is counted in all abstracts (*abstractCount*).

This results in a list with tags that are related to the related DBpedia resources.

To demonstrate the results, the ranked tag lists are compared again for Cristiano Ronaldo, Amsterdam and Arctic Monkeys. Since most of the Kaartje2go tags are Dutch, the Dutch abstracts are used to find related tags. The tag list, ranked by occurrences can be found in Table 10.

Rank #	Cristiano Ronaldo	Amsterdam	Arctic Monkeys
1	Madrid (54)	Amsterdam (32)	Music (68)
2	Season (51)	Countries (23)	Rock (53)
3	Portugal (43)	Water (21)	Guitar (34)
4	Contract (37)	Utrecht (13)	England (23)
5	Lissabon (34)	Ministry (10)	Metal (13)
6	Barcelona (28)	Haarlem (9)	River (13)
7	Football (28)	Politics (9)	London (11)
8	Euro (18)	Spain (6)	Word (11)
9	Countries (18)	France (6)	Wave (9)
10	Spanish (16)	House (6)	English (8)

TABLE 10: FOUND TAGS. TRANSLATED FROM DUTCH FOR THE READER'S CONVENIENCE.

4.3.4 RECOMMENDATION CREATOR

This section describes the recommendation creator. This component in IBRS uses the tags that were found by the 'Tag Extractor' to create items from the tagged item set.

The results presented in Table 11 shows that the 'Tag Extractor' was able to find related tags from the tag set based on the user's interests. However, some tags are found that does not seem related to the user's interests. For example, the tag "Countries" occurs twice in Table 10 from totally different items. Since these tags are very general they tend to have a high term frequency. In the third chapter of this thesis we found that TF-IDF can help ranking the importance of words, or tags in our case. We used a variation of TF weight, namely log normalization.

For each tag that was mapped to a DBpedia resource, the following metadata was added:

- *inlinks*: The number of links to the DBpedia resource of the tag. Together with *abstractCount* this gives an indication of the importance of a tag and
- *tagCount*: The term frequency of this tag in the complete DBpedia dataset.

We use the following formula to compute a ranking score for each tag:

$$Ranking\ score\ = \frac{abstractCount\ *\ log(inlinks)}{log(tagCount)}$$

Applying this formula to the complete set of found tags, results in a ranked list of related tags.

The high-ranked tags have the highests relevance the user. To improve precision, the top-n tags can be used to create recommendations. Looking at sample data, we found that using the top-10 tags works best for IBRS.

The next step is to match these tags to items in IBRS so recommendations can be created. In IBRS we implemented two methods for selecting items, namely a weighted-random-based variant and weighted-tag-based variant.

In the weighted-random-based variant, every recommendation is based on a single tag. This tag is randomly selected using weights based on the tag's ranking score. The following query is used to select an item from the dataset, using a single tag:

```
   SELECT card.id, card.title, card.url
     FROM cards card
       JOIN abstract_items_tags ait ON ait.abstract_item_id = card.abstract_item_id
       JOIN tags as tag ON tag.id = ait.tag_id
        AND ait.type = 'card'
        AND tag.name = 'randomTag1'
ORDER BY rand()
      LIMIT 1
```

The other variant for selecting items in IBRS, is based on the number of matched tags. Items that are associated with more related tags are ranked higher. To select the postcards, we use the following query.

```
    SELECT card.*, group_concat(t.name_eng) as tags
      FROM cards card
        JOIN abstract_items_tags ait
          ON ait.abstract_item_id = card.id
         AND ait.type = 'card'
        JOIN tags t
          ON ait.tag_id = t.id
    WHERE t.name IN ('tag1', 'tag2', 'tag3)
 GROUP BY card.id
 ORDER BY count(*) DESC, rand() # First order by number of tags, then random
      LIMIT 1
```

CHAPTER 5:

EVALUATION

This chapter describes how the performance of IBRS is evaluated. Section 5.1 describes the two baseline system there are used. In Section 5.2 a description of the online evaluation tool is given. The results of the online evaluation sessions are presented in Section 5.3.

5.1 BASELINE SYSTEMS

Two online evaluation sessions were done to test IBRS. In these evaluation experiments different aspects were tested, namely the usefulness of DBpedia and the performance of IBRS. For both the evaluation sessions we used different baseline RS. In the remainder of this section these baseline RS are explained.

FacebookOnly:

Using the information in DBpedia to find the abstracts of related items results in a long text with unstructured information. In the first evaluation session the usefulness of all this DBpedia information is evaluated. In order to do this IBRS is compared to a baseline system that does not use the DBpedia data.

This baseline is called *FacebookOnly*. This RS is based on IBRS but the Facebook Mapper and the DBpedia Explorer are removed. The descriptions of the Facebook pages are used as input for the Tag Selector. This way DBpedia is not used and the recommendations are solely based on the descriptions of the Facebook pages. In this evaluation experiment, the recommendations are created using the weighted-random-based item selector.

InvertedIBRS:

The second baseline system is also based on IBRS and is called *InvertedIBRS*. Using this baseline system, we want to verify how good IBRS is at predicting useful items. Normal IBRS takes the most related tags and finds with items have these tags. *InvertedIBRS* on the other hand, finds the items that don't have an associated with a related tag. Therefore, it recommends items that we believe to be less relevant for the user. In this evaluation experiment, the recommendation are created using the weighted-tag-based item selector.

5.2 WEBSITE

We created a web application to evaluate the performance of IBRS for the postcards from Kaartje2go and the holiday homes from the online holiday home portal EuroCottage. This application was publicly accessible for users with a direct URL, and enabled us to test the system's performance in comparison to the two baseline systems.

Welcome screen

The welcome screen, as seen in Figure 21, shows a welcome and a description of the tasks the users should perform. When the user clicks the 'Get started!' button he will be redirected to Facebook to grant us the needed permissions to view their likes.

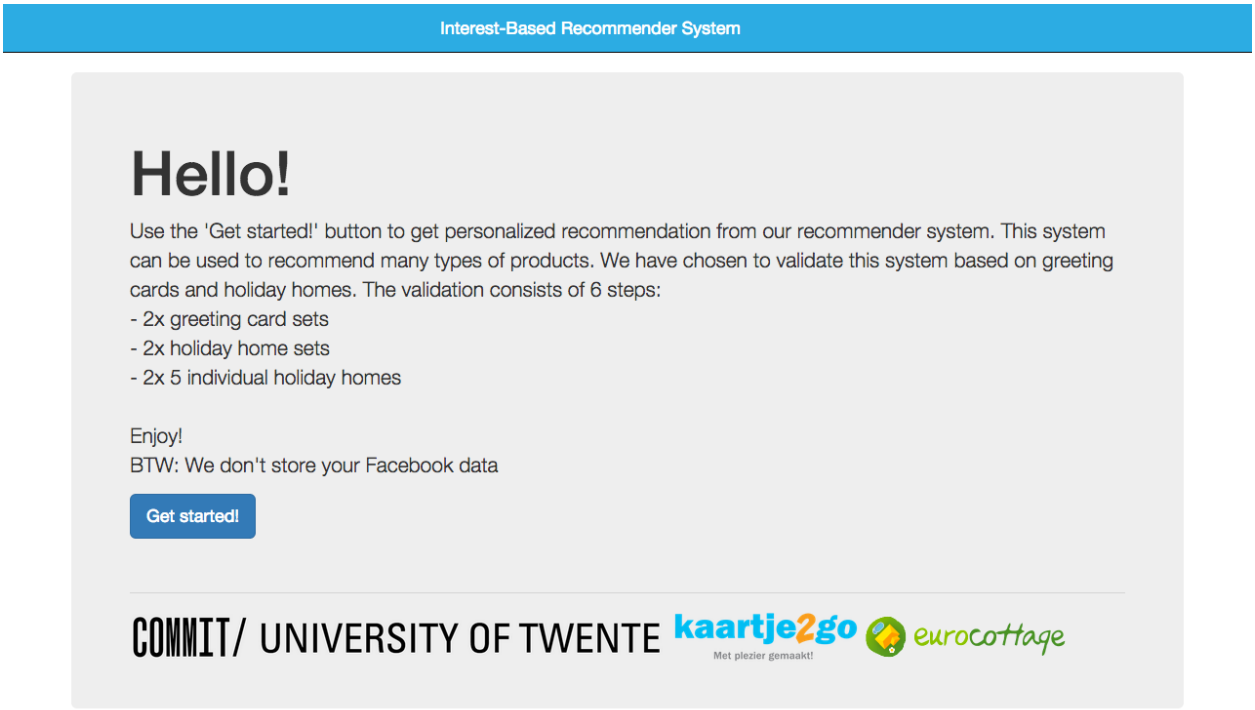


FIGURE 21: WELCOME SCREEN OF THE EVALUATION WEBSITE.

When the user has granted us to view their likes, the likes are queried and are inserted into IBRS to do its magic. This process may take a while, depending on the likes of the user. It takes about 5 to 10 seconds to complete.

First evaluation screen: the cards

In the next steps the real evaluation process begins. First, the user is asked to evaluate the cards. Two sets of cards are presented to the user, as seen in Figure 22. One set of cards is created using the baseline system and the other set is created with IBRS. The position of these baseline system is randomly chosen. The users are asked to select the set of cards that they like best twice.

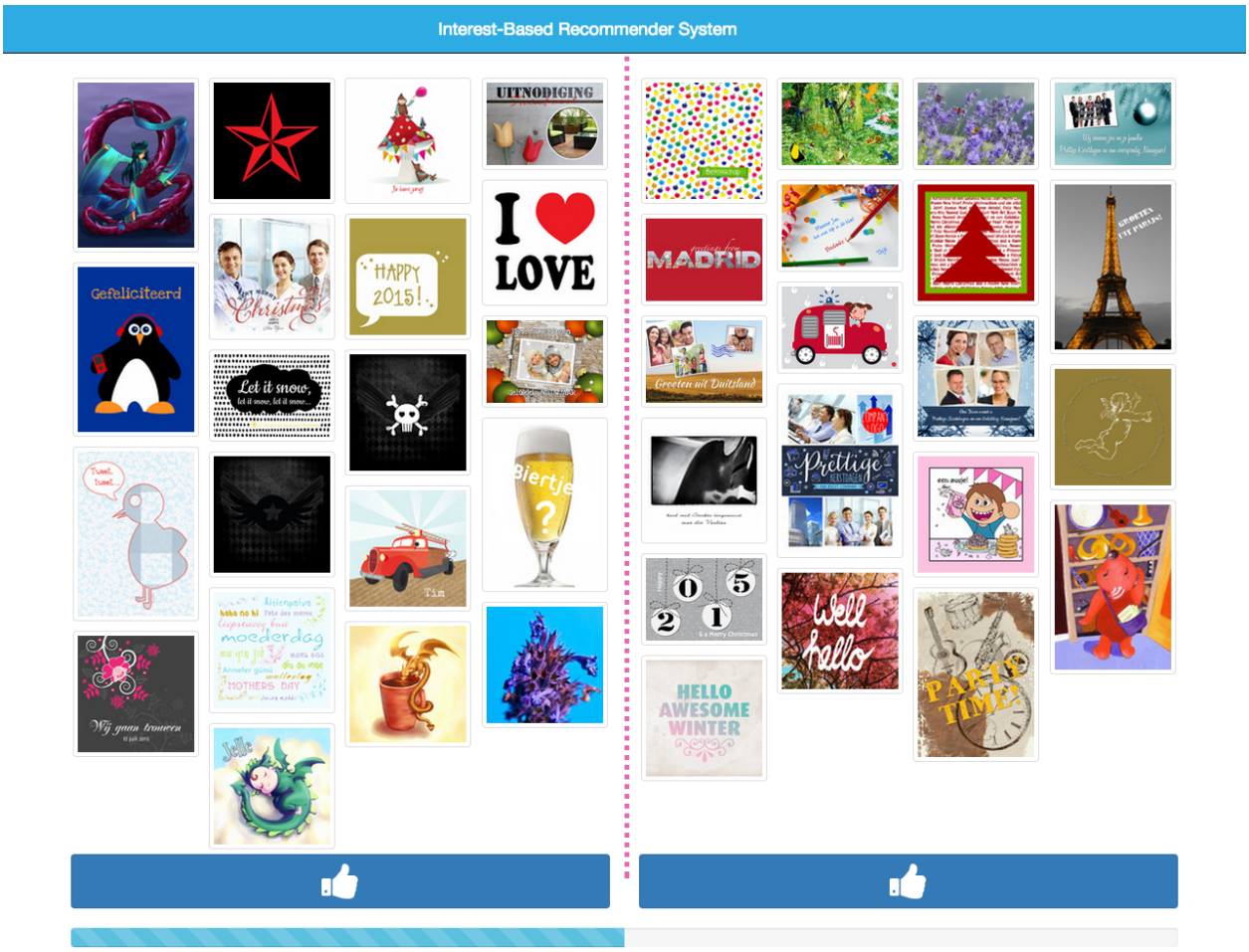


FIGURE 22: SCREENSHOT OF EVALUATION PAGE FOR CARDS.

Second evaluation screen: the holiday houses

After the user has selected the best card set twice, he is redirected to the evaluation page for holiday houses. This page is very similar the evaluation page for cards. The user can select which set of holiday houses he likes the most. The only difference is the list of associated tags that are shown alongside the description of the property, as seen in Figure 23.

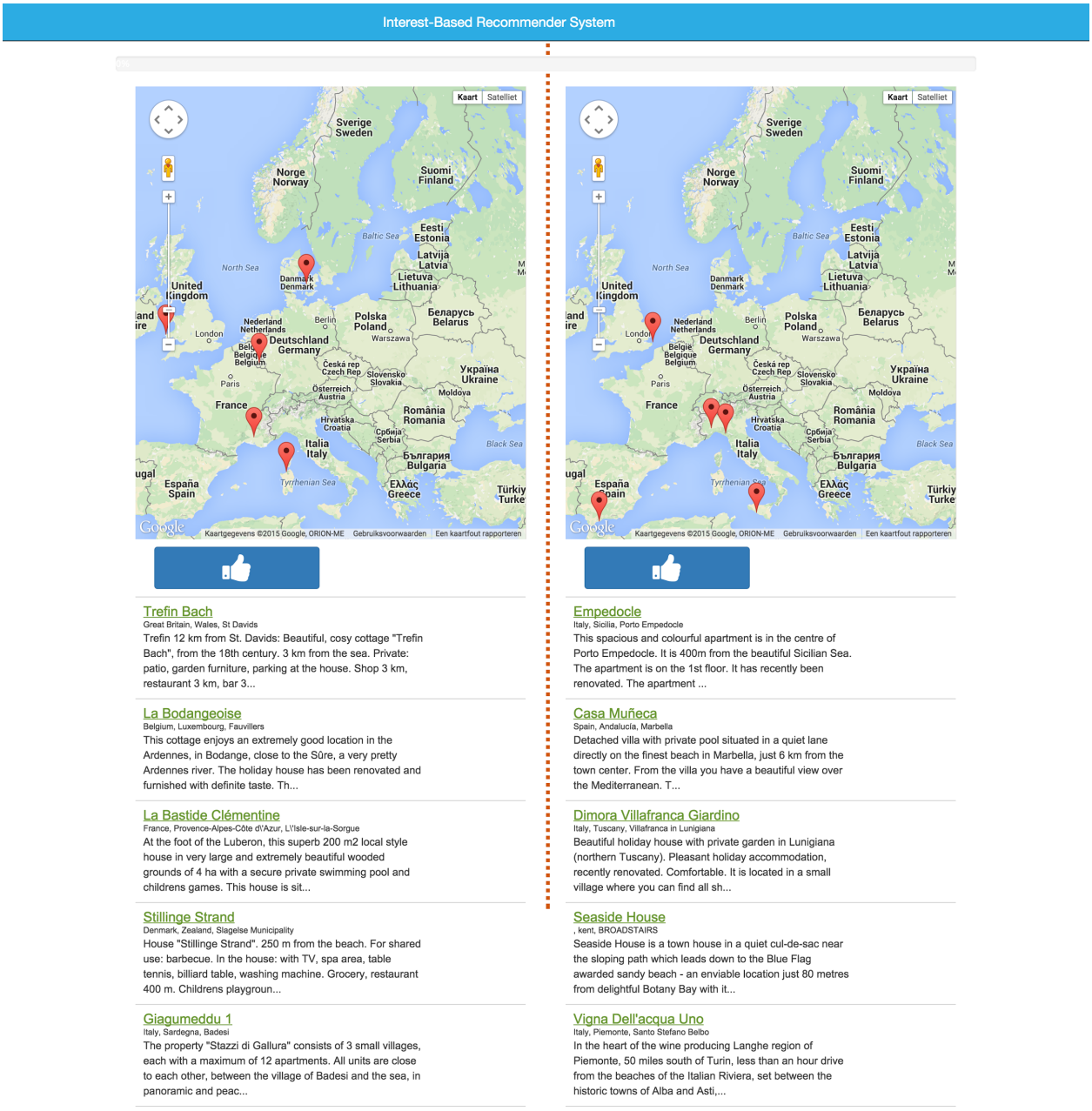


FIGURE 23: SCREENSHOT FOR THE EVALUATION OF THE RECOMMENDATIONS FOR HOLIDAY HOMES.

Third evaluation screen: Rating the recommendations

After the user has picked the best holidays houses set twice he is redirect to another type of evaluation page for these properties. On this page, as seen in Figure 24, he can rate how good the recommendation is interesting to him using the Likert 5-step rating scale:

1. Strongly disagree;
2. Disagree;
3. Neither agree nor disagree;
4. Agree, or;
5. Strongly agree.

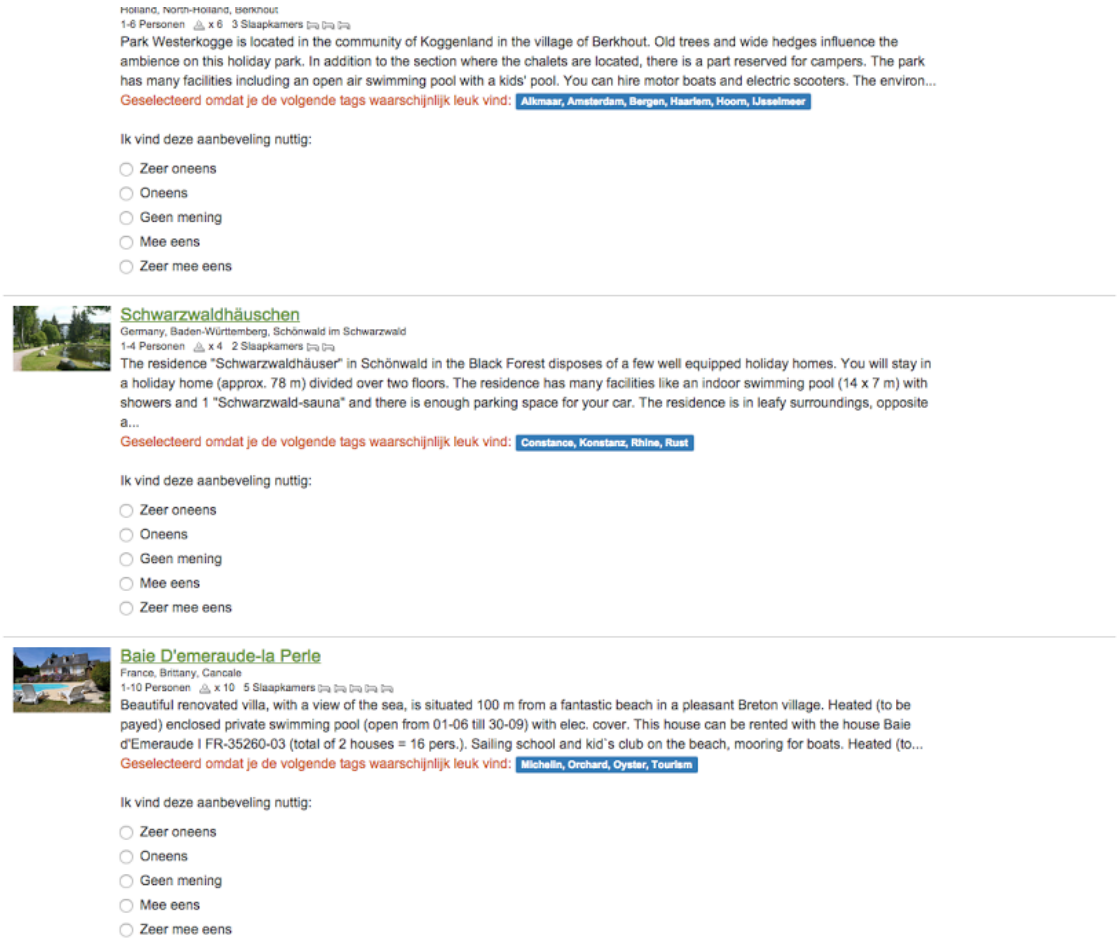


FIGURE 24: SCREENSHOT FOR THE EVALUATION OF THE RECOMMENDATIONS FOR HOLIDAY HOMES USING THE LIKERT SCALE.

We conducted two evaluation sessions. The screenshots that are shown here are taken from the first evaluation session. The second evaluation session was very similar, but there the user could only choose between two cards or properties. This task was repeated 10 times. Since these number where configurable in the application we could easily create the second evaluation session.

5.3 RESULTS

This section describes the actual results for the two evaluation session where we compared IBRS to two baseline systems, FacebookOnly and InvertedIBRS.

First the evaluation session with FacebookOnly was done. Based on the results from FacebookOnly we did another evaluation session with InvertedIBRS one month later. Therefore, we explain the results for both baseline system separately in the following sections.

5.3.1 RESULTS WITH FACEBOOKONLY

71 people have helped evaluating IBRS in the first evaluation round by selecting the sets that they liked best. These users have made 111 votes for the cards and 109 votes for the properties. Contrary to our expectations, the results were slightly in favor of the baseline method, which uses only the Facebook descriptions.

For the cards, the users selected 60 times the baseline system and 51 times IBRS for creating the best set of cards. The baseline method performs 17.65% better. The difference between the two systems was smaller for the holiday homes. The baseline system was selected 50 times as the best system and IBRS was selected 59 times as the best system.

Figure 25 shows an overview of the results for the cards per vote. In the first evaluation page the set created by the baseline system was by far the most selected set with 32 vs. 24 votes. On the second evaluation page the baseline system was still the best but the difference was much smaller with 27 vs 26 votes.

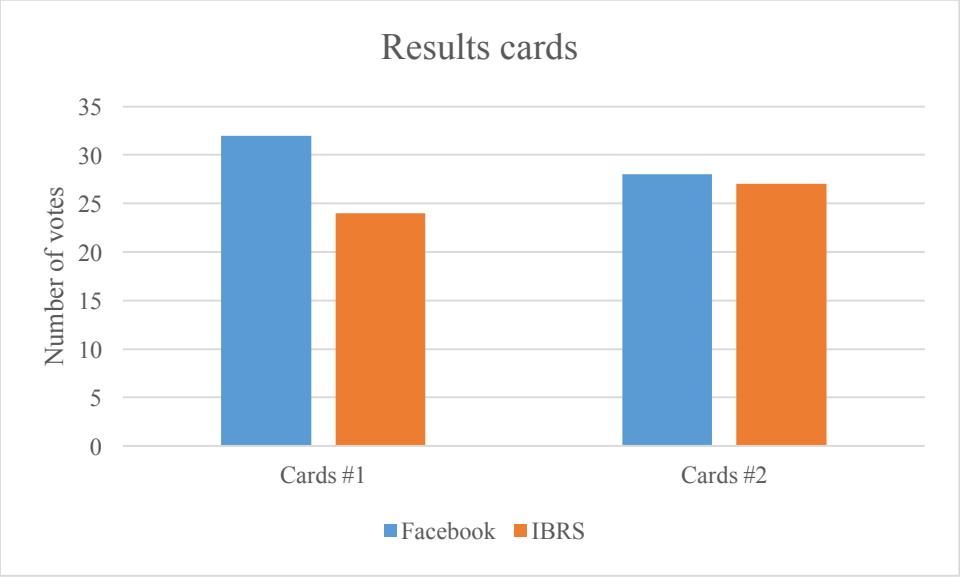


FIGURE 25: OVERVIEW OF THE RESULTS FOR THE CARDS FOR THE TWO EVALUATION PAGES.

Figure 26 contains the voting results for the holiday houses. The first time properties picked by the IBRS received the most votes, with 32 vs. 23 votes. On second evaluation page both sets were selected the same number of times, namely 25.

FIGURE 26: OVERVIEW OF THE RESULTS FOR THE HOLIDAY HOUSES FOR THE TWO EVALUATION PAGES.

The users were overall positive about the recommendations. Through informal interviews with the test users, we received the feedback that they really had the feeling the recommendations were personalized. One test user found the evaluation method not very user-friendly, since they didn't know how to select cards. Another problem was the recommendations themselves, since users didn't like all recommended items in a set, and had no way of providing feedback for this in this set-wise comparison.

5.3.2 Results with InvertedIBRS

After we found that DBpedia did not contribute much in finding related items, we wanted to evaluate the effectiveness of IBRS and compared IBRS with InvertedIBRS.

In total 759 votes are made in the second evaluation session, 414 for the cards and 345 for the holiday houses. The users choose IBRS more often with 418 votes for IBRS and 341 for InvertedIBRS, see Figure 27.

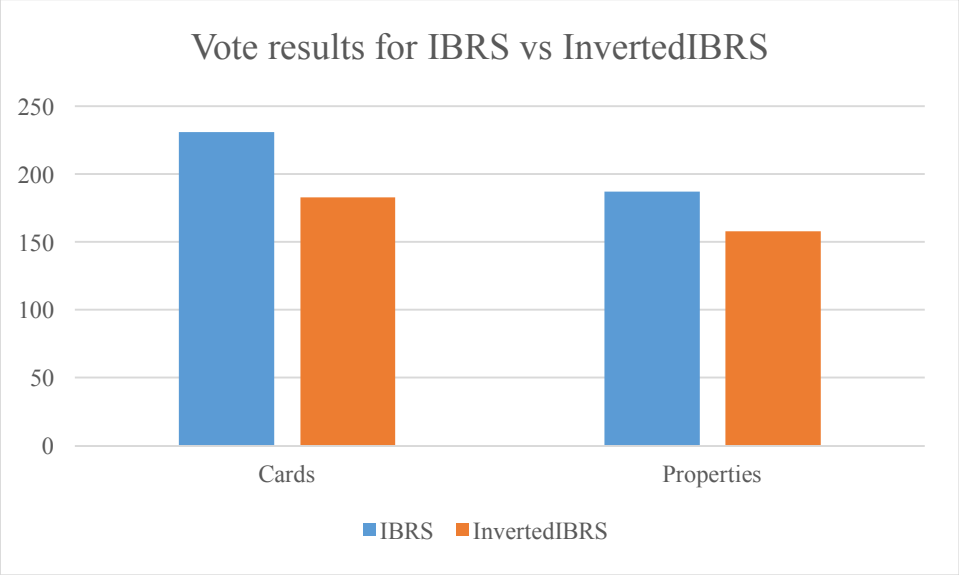


Figure 27: Votes results for IBRS vs InvertedIBRS

Using user session IDs, we were able to determine how many people selected IBRS more often than *InvertedIBRS*. Most users liked IBRS better than *InvertedIBRS*. This difference is clearer for the properties, see Figure 28.

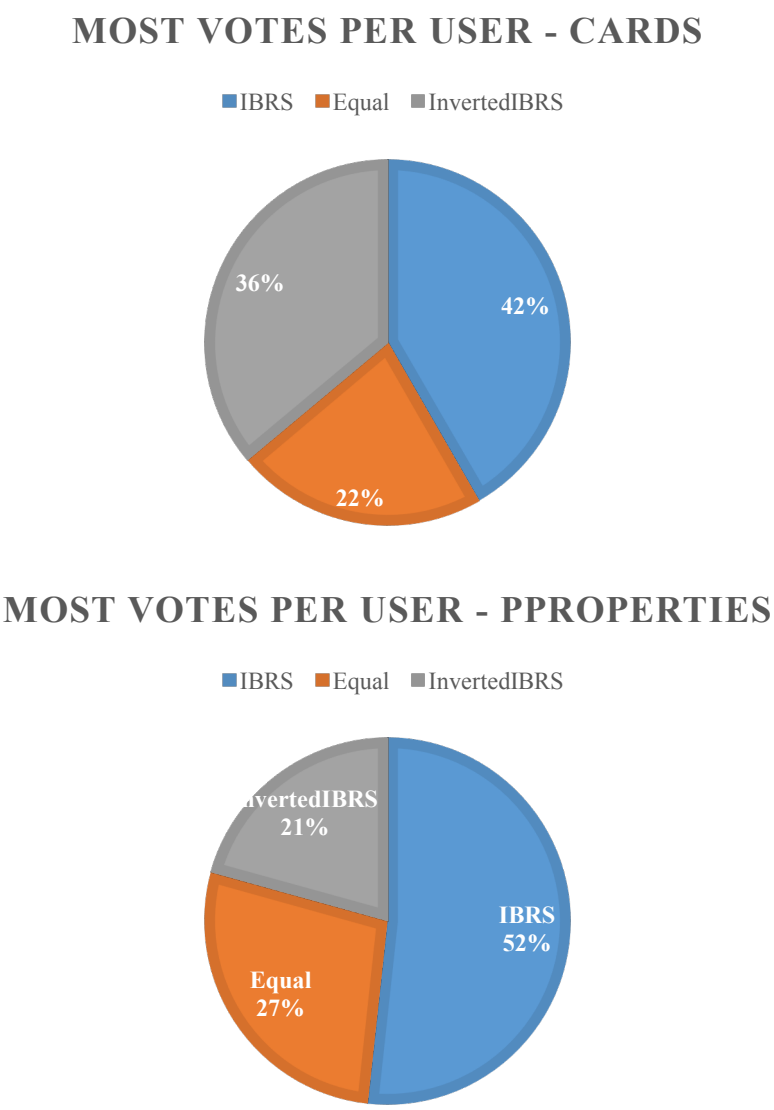


FIGURE 28: VOTE RESULTS WITH MOST VOTES PER USER.

The users were also asked to give a rating for the recommendations in the last step. These recommendations were all made by IBRS. The first session received 465 votes, while the second session received 218 votes. Changing the ranking method in the second evaluation session did improve the average rating, especially for the recommendations without explanation, see Figure 29.

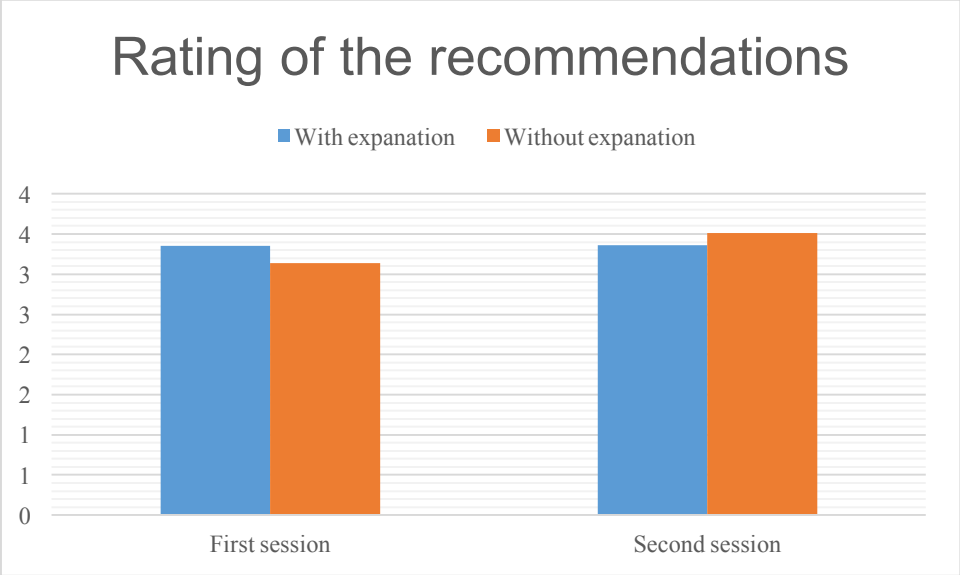


FIGURE 29: VOTES RESULTS WITH RATING OF RECOMMENDATIONS.

5.4 VALIDATION RESULTS

In this section we will discuss the results from our validation experiment and explain some (unexpected) results.

Facebook descriptions vs DBpedia

Based on the results, we can conclude that the test users did not like the recommendations better that are created using the information in DBpedia. This result was quite unexpected, since DBpedia contains much more information than the descriptions from the Facebook pages. We assumed that more information would result in better recommendations, since more tags can be found.

Looking at the generated tags and recommendations, this effect can best be explained using the definitions of precision and recall. Since all the abstracts from the related items in DBpedia contains much more information, more tags are found, and the recall increases. On the other hand, the change of finding general tags increases, and therefore more general recommendations are created. So as more (general) tags are found, more filtering should be applied to the found tags.

Another drawback of using DBpedia is that the user's interests must first be mapped to resources inside the DBpedia graph. We found that 19% of the Facebook likes could be mapped to DBpedia resources, while using Facebook descriptions, all Facebook pages can be used to extract useful tags.

Recommendation selector: Weighted-random vs. weighted-tags

During the validation experiments, two different recommendation selectors were tested. The first variant uses only a single tag to find recommended items and in second variant the recommended items are sorted based on the number of associated tags.

Based on the votes from the test users for the recommendations, we can conclude that the variant based on the number of matched tags performed slightly better. If more related tags are associated with an item, this item will be more interesting for the user.

The dataset used in the validation experiments contain a lot of items that are associated with only generic tags. Another advantage of the weighed-tag approach is that these items will be recommended less frequently. Since other items with more useful tags will be ranked higher.

IBRS vs InvertedIBRS

Based on the validation results we can conclude that IBRS can, in most cases, successfully determine if a user likes an item, based in the user's interests. 418 times the users preferred the recommendations by IBRS versus 341 votes where the users preferred the recommendations from InvertedIBRS.

This shows IBRS has the potential to create useful recommendations. Still, a lot of votes are done for items that IBRS couldn't map to the user's social interests. This exposes a limitation of IBRS, since not all aspects needed for decision making can be retrieved from a social profile. There is often still some domain knowledge needed for recommending items. For example, postcards can be selected based on colors or design and holidays homes can be selected based on price, number of bedrooms or central location.

CHAPTER 6:

CONCLUSIONS

In this thesis we presented Interest-Based Recommender System (IBRS): a domain-independent, language-independent, knowledge-based RS.

First, we looked at current RS's and their limitations. We found how a lot of structured information can be accessed via open knowledge-bases like DBpedia. This motivated us to investigate the potential of a generic knowledge-based RS. We use DBpedia to detect a user's interests from items related to his Facebook likes.

We evaluated IBRS in two domains, postcards and holiday houses in two different evaluation sessions. In the first evaluation session we tested IBRS versus a baseline RS that is solely based on the descriptions from the Facebook pages. From the evaluations we learned that with or without explanation, people tended to review the IBRS recommendations positively. However, we could not detect a significant difference in performance from the interest extrapolation through DBpedia. The baseline system scored slightly better in the evaluation for the postcards and IBRS scored a bit better for the holiday houses.

In the second evaluation session we changed two aspects. First of all, we altered the ranking method to a method that ranked based on the number of matched tags. Secondly, we changed our baseline to an inverse of IBRS, which returned those items with exactly 0 matched tags. The new version of IBRS was well received by the test users, with over, with most people preferring IBRS results over Inverted IBRS.

Overall, the results look very promising. IBRS can generate recommendations without any domain-knowledge in any language. However, we still see several possibilities to improve our system. Selecting the right tags for recommendation extraction is a crucial step in a tag-based RS. Comparing several tag ranking approaches with a test user group may lead to even better results. However, we could not expect our test users to participate in too many validation rounds. Therefore, such a comparison will need to be done on an existing dataset, for example from watched movies or purchased greeting cards.

Furthermore, IBRS can be used for other applications. Currently it is used as a system that recommends items, based on the user's interests. We found that it can also be used as a query expansion system, without using the user's social media profile. In the Kaartje2go dataset some football postcards were tagged with names of popular soccer clubs. So if a user searches for a football club he finds postcards about football. Using the information in IBRS searches for other football clubs may also results in postcard about football.

Over the years lots of people have created and updated social profiles where they reveal what they do and possibly do not like. IBRS uses this information to help customers finding related content. While users do not publish all their personal interests online and not always like related items, using IBRS in existing RSs may give good pointers for selecting the right content, like for example greeting cards.

ACKNOWLEDGEMENTS

This publication was supported by the Dutch national program COMMIT/.

BIBLIOGRAPHY

[1] CBS. (2014, April) CBS. [Online]. http://statline.cbs.nl/Statweb/publication/?DM=SLNL&PA=71098NED&D1=114-133&D2=0-2&D3=a&HDR=G1&STB=T,G2&VW=T

[2] J. S. Armstrong (Ed.), *Principles of forecasting: a handbook for researchers and practitioners*.: Springer Science & Business Media, 2001, vol. 30.

[3] B. N., Albert, I., Lam, S. K., Konstan, J. A., & Riedl, J. Miller, "MovieLens unplugged: experiences with an occasionally connected recommender system," *Proceedings of the 8th international conference on Intelligent user interfaces*, pp. 263-266, January 2003.

[4] J. Nolin, "Learning Technologies That Are Not Meant for Learning: a critical discussion of learning objects," *Human IT*, vol. 11, no. 2, 2013.

[5] X., Basilico, J. Amatriain. (2012, April) Netflix. [Online]. http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html

[6] J., & Lanning, S. Bennett, "The Netflix Prize," *Proc. KDD-Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. 2007, p. 35, August 2007.

[7] R. Burke, "Hybrid web recommender systems," *The adaptive web*, pp. 377-408, 2007.

[8] J., Ortega, F., Hernando, A., & Gutiérrez, A. Bobadilla, "Recommender systems survey," *Knowledge-Based Systems*, vol. 46, pp. 109-132, 2013.

[9] P., De Gemmis, M., & Semeraro, G. Lops, "Content-based recommender systems: State of the art and trends," *Recommender systems handbook*, pp. 73-105, 2011.

[10] V., & Capra, L. Zanardi, "Social ranking: uncovering relevant content using tag-based recommender systems," *Proceedings of the 2008 ACM conference on Recommender systems*, pp. 51-58, October 2008.

[11] D. Oosterveer. (2014, January) Marketingfacts. [Online]. http://www.marketingfacts.nl/berichten/nationale-social-media-onderzoek-2014

[12] F., Rokach, L., & Shapira, B. Ricci, *Introduction to Recommender Systems Handbook*.: Springer, 2011.

[13] M. J., & Billsus, D. Pazzani, "Content-based recommendation systems," *The Adaptive Web*, pp. 325-341, 2007.

[14] G., & Tuzhilin, A. Adomavicius, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 17, no. 6, pp. 734-749, 2005.

[15] J. S., Heckerman, D., & Kadie, C. Breese, "Empirical analysis of predictive algorithms for collaborative filtering," *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pp. 43-52, July 1998.

[16] X., & Khoshgoftaar, T. M. Su, "A survey of collaborative filtering techniques," *Advances in artificial intelligence*, vol. 2009, p. 4, 2009.

[17] B., Karypis, G., Konstan, J., & Riedl, J. Sarwar, "Item-based collaborative filtering recommendation algorithms," *Proceedings of the 10th international conference on World Wide Web*, pp. 285-295, April 2001.

[18] J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. Herlocker, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 5-53, 2004.

[19] K., & Van Stam, W. Ali, "TiVo: making show recommendations using a distributed collaborative filtering architecture," *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 394-401, August 2004.

[20] R., & Sajja, P. Akerkar, *Knowledge-based systems*.: Jones & Bartlett Publishers, 2010.

[21] M. H., Hong, J. H., & Cho, S. B. Park, "Location-based recommendation system using bayesian user's preference model in mobile devices," *Ubiquitous Intelligence and Computing*, pp. 1130-1139, 2007.

[22] T. H., Oh, K. J., & Han, I. Roh, "The collaborative filtering recommendation based on SOM cluster-indexing CBR," *Expert Systems with Applications*, vol. 25, no. 3, pp. 413-423, 2003.

[23] R. R. Yager, "Fuzzy logic methods in recommender systems," *Fuzzy Sets and Systems*, vol. 136, no. 2, pp. 133-149, 2003.

[24] J., & Li, X. Zhong, "Unified collaborative filtering model based on combination of latent features," *Expert Systems with Applications*, vol. 37, no. 8, pp. 5666-5672, 2010.

[25] X., Xia, Y., & Zhu, Q. Luo, "Incremental collaborative filtering recommender based on regularized matrix factorization," *Knowledge-Based Systems*, vol. 27, pp. 271-280, 2012.

[26] B. Krulwich, "Lifestyle finder: Intelligent user profiling using large-scale demographic data," *AI magazine*, vol. 18, no. 2, p. 37, 1997.

[27] R. D., Hammond, K. J., & Yound, B. C. Burke, "The FindMe approach to assisted browsing," *IEEE Expert*, vol. 12, no. 4, pp. 32-40, 1997.

[28] R. Burke, "The Wasabi Personal Shopper: a case-based recommender system," *Proceedings of the 11th national conference on innovative applications of artificial intelligence*, pp. 844-849, 1999.

[29] N., & Ibrahim, D. Cavus, "m-Learning: An experiment in using SMS to support learning new English language words," *British journal of educational technology*, vol. 40, no. 1, pp. 78-91, 2009.

[30] C., Tejeda-Lorente, A., Martínez, M. A., & Herrera-Viedma, E. Porcel, "A hybrid recommender system for the selective dissemination of research resources in a technology transfer office," *Information Sciences*, vol. 184, no. 1, pp. 1-19, 2012.

[31] M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., & Sartin, M. Claypool, "Combining content-based and collaborative filters in an online newspaper," *Proceedings of ACM SIGIR workshop on recommender systems*, vol. 60, 1999.

[32] M. J. Pazzani, "A framework for collaborative, content-based and demographic filtering," *Artificial Intelligence Review*, vol. 13, no. 5-6, pp. 393-408, 1999.

[33] I., & Nicholas, C. Soboroff, "Combining Content and Collaboration in Text Filtering," *Proceedings of the IJCAI*, pp. 86-91, 1999.

[34] B., Karypis, G., Konstan, J., & Riedl, J. Sarwar, "Incremental singular value decomposition algorithms for highly scalable recommender systems," *Fifth International Conference on Computer and Information Science*, pp. 27-28, 2002.

[35] B. M., Konstan, J. A., & Riedl, J. Sarwar, "Distributed Recommender Systems for Internet Commerce," 2005.

[36] B., Han, P., Yang, F., & Shen, R. Xie, "An efficient neighbor searching scheme of distributed collaborative filtering on p2p overlay network," *Database and Expert Systems Applications*, pp. 141-150, 2004.

[37] P., Xie, B., Yang, F., & Shen, R. Han, "A scalable P2P recommender system based on distributed collaborative filtering," *Expert systems with applications*, vol. 27, no. 2, pp. 203-210, 2004.

[38] T., & Merugu, S. George, "A scalable collaborative filtering framework based on co-clustering," *Data Mining, Fifth IEEE International Conference on*, p. 4, 2005.

[39] M., Rousidis, I., Plexousakis, D., & Theoharopoulos, E. Papagelis, "Incremental collaborative filtering for highly-scalable recommendation algorithms," *Foundations of Intelligent Systems*, pp. 553-561, 2005.

[40] I., Cantador, I., Kaminskas, M., & Ricci, F. Fernández-Tobías, "A generic semantic-based framework for cross-domain recommendation," *Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, pp. 25-32, 2011.

[41] A., & Burke, R. Felfernig, "Constraint-based recommender systems: technologies and research issues," *Proceedings of the 10th international conference on Electronic commerce*, p. 3, 2008.

[42] C., Heath, T., & Berners-Lee, T. Bizer, "Linked Data - The Story So Far," *International Journal on Semantic Web and Information Systems*, vol. 5, no. 3, pp. 1-22, 2009.

[43] B. McBride, "The resource description framework (RDF) and its vocabulary description language RDFS," *Handbook on ontologies*, pp. 51-65, 2004.

[44] E. Miller, "An introduction to the resource description framework," *Bulletin of the American Society for Information Science and Technology*, vol. 25, no. 1, pp. 15-19, 1998.

[45] M., Bizer, C., Jentzsch, A., Cyganiak, R. Schmachtenberg. (2014, August) The Linking Open Data cloud diagram. Image.

[46] S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., & Ives, Z. Auer, "Dbpedia: A nucleus for a web of open data," *Proceedings of the 6th International The Semantic Web and 2Nd Asian Conference on Asian Semantic Web Conference*, pp. 722-735, 2007.

[47] DBpedia. DBpedia. [Online]. http://wiki.dbpedia.org/

[48] G., & Widom, J. Jeh, "SimRank: a measure of structural-context similarity," *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 538-543, 2002.

[49] J., Sun, Y., Yan, X., & Yu, P. S. Han, "Mining heterogeneous information networks," *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, p. 5, 2010.

[50] C., Kong, X., Yu, P. S., Xie, S., & Wu, B. Shi, "Relevance search in heterogeneous networks," *Proceedings of the 15th international conference on extending database technology*, pp. 180-191, 2012.

[51] Y., Han, J., Yan, X., Yu, P. S., & Wu, T. Sun, "Pathsim: Meta path-based top-k similarity search in heterogeneous information networks," *PVLDB*, vol. 4, no. 11, pp. 992-1003, 2011.

[52] P., Nanopoulos, A., & Manolopoulos, Y. Symeonidis, "MoviExplain: a recommender system with explanations," *Proceedings of the third ACM conference on Recommender systems*, pp. 317-320, 2009.

[53] R., Di Noia, T., Ostuni, V. C., & Ragone, A. Mirizzi, "Linked Open Data for content-based recommender systems," *Proceedings of the 8th International Conference on Semantic Systems*, pp. 1-8, 2012.

[54] C., Zhou, C., Kong, X., Yu, P. S., Liu, G., & Wang, B. Shi, "HeteRecom: a semantic-based recommendation system in heterogeneous networks," *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1552-1555, 2012.

[55] C., Kong, X., Huang, Y., Yu, P. S., & Wu, B. Shi, "Hetesim: A general framework for relevance measure in heterogeneous networks," *Knowledge and Data Engineering*, vol. 26, no. 10, pp. 2479-2492, 2014.

[56] A. Passant, "dbrec—music recommendations using DBpedia," *The Semantic Web–ISWC 2010*, pp. 209-224, 2010.

[57] V. C., Di Noia, T., Di Sciascio, E., & Mirizzi, R. Ostuni, "Top-n recommendations from implicit feedback leveraging linked open data," *Proceedings of the 7th ACM conference on Recommender systems*, pp. 85-92, 2013.

[58] R., Ragone, A., Di Noia, T Mirizzi, "Ranking the Linked Data: The Case of DBpedia," *In 10th Int. Conf. on Web Engineering (ICWE}*, 2010.

[59] P., De Gemmis, M., & Semeraro, G. Lops, "Content-based recommender systems: State of the art and trends," *Recommender systems handbook*, pp. 73-105, 2011.