CBRecSys 2015. New Trends on Content-Based Recommender Systems

Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015)

Bogers, Toine; Koolen, Marijn

Publication date:
2015

Document Version
Publisher's PDF, also known as Version of record

Link to publication from Aalborg University

Citation for published version (APA):
Bogers, T., & Koolen, M. (Eds.) (2015). *CBRecSys 2015. New Trends on Content-Based Recommender Systems: Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015)*. CEUR Workshop Proceedings. CEUR Workshop Proceedings Vol. 1448 http://ceur-ws.org/Vol-1448/

Workshop proceedings

CBRecSys 2015

2nd Workshop on New Trends in Content-based Recommender Systems

September 20, 2015
RecSys 2015, Vienna, Austria

Edited by
Toine Bogers and Marijn Koolen

Preface

While content-based recommendation has been applied successfully in many different domains, it has not seen the same level of attention as collaborative filtering techniques have. In recent years, competitions like the Netflix Prize, CAMRA, and the Yahoo! Music KDD Cup 2011 have spurred on advances in collaborative filtering and how to utilize ratings and usage data. However, there are many domains where content and metadata play a key role, either *in addition to* or *instead of* ratings and implicit usage data. For some domains, such as movies the relationship between content and usage data has seen thorough investigation already, but for many other domains, such as books, news, scientific articles, and Web pages we do not know *if* and *how* these data sources should be combined to provided the best recommendation performance.

The CBRecSys workshop series aims to address this by providing a dedicated venue for papers dedicated to all aspects of content-based recommendation. The first edition in Silicon Valley in 2014 was a big success with over 60 attendees and 16 submissions.

For the second edition, CBRecSys 2015, we once again issued a Call for Papers asking for submissions of novel research papers (both long and short) addressing recommendation in domains where textual content is abundant (e.g., books, news, scientific articles, jobs, educational resources, Web pages, etc.) as well as dedicated comparisons of content-based techniques with collaborative filtering in different domains. Other relevant topics included opinion mining for text/book recommendation, semantic recommendation, content-based recommendation to alleviate cold-start problems, as well as serendipity, diversity and cross-domain recommendation.

Each submission was received by three members of the program committee consisting of experts in the field of recommender systems and information retrieval. We selected 6 long papers and 2 short papers from the 12 submissions for presentation at the workshop. We are also happy to have professor Frank Hopfgartner of the University of Glasgow give a keynote presentation on capturing user interests for content-based recommendation.

We thank all PC members, our keynote speaker as well as authors of accepted papers for making CBRecSys 2015 possible. We hope you will enjoy the workshop!

Toine Bogers & Marijn Koolen

Table of Contents

Organization

Workshop organizers

- Toine Bogers, *Aalborg University Copenhagen, Denmark*
- Marijn Koolen, *University of Amsterdam, the Netherlands*

Program committee

- Robin Burke, *DePaul University, USA*
- Iván Cantador, *Universidad Autónoma de Madrid, Spain*
- Federica Cena, *Universita' degli Studi di Torino, Italy*
- Paolo Cremonesi, *Politecnico di Milano, Italy*
- Marco De Gemmis, *University of Bari "Aldo Moro", Italy*
- Ernesto W. De Luca, *Potsdam University of Applied Sciences, Germany*
- Tommaso Di Noia, *Politecnico di Bari, Italy*
- Peter Dolog, *Aalborg University, Denmark*
- Soude Fazeli, *Open University*
- Juan F. Huete, *Universidad de Granada, Spain*
- Jaap Kamps, *University of Amsterdam*
- Birger Larsen, *Aalborg University Copenhagen, Denmark*
- Babak Loni, *Delft University of Technology*
- Pasquale Lops, *University of Bari "Aldo Moro", Italy*
- Cataldo Musto, *University of Bari "Aldo Moro"*
- Casper Petersen, *University of Copenhagen*
- Shaghayegh Sahebi, *University of Pittsburgh*
- Alan Said, *Recorded Future, Sweden*
- Giovanni Semeraro, *University of Bari "Aldo Moro", Italy*
- Nafiseh Shahib, *Norwegian University of Science and Technology*
- Marko Tkalčič,*Johannes Kepler University, Austria*
- Bei Yu, *Syracuse University*

Capturing User Interests for Content-based Recommendations

Frank Hopfgartner
Humanities Advanced Technology and Information Institute
University of Glasgow
Glasgow, UK
frank.hopfgartner@glasgow.ac.uk

ABSTRACT

Nowadays, most information filtering systems provide recommendations by either building a model of the users' past behavior, referred to as collaborative filtering, or by identifying items with similar properties, referred to as content-based recommendation.

In this presentation, I will present various use cases and scenarios where recommendations are provided based on a preceding content analysis. The use cases will illustrate both strengths and weaknesses of content-based recommendation.

CBRecSys 2015, September 20, 2015, Vienna, Austria.

Generic knowledge-based analysis of social media for recommendations

Victor de Graaff
Dept. of Computer Science
University of Twente
Enschede, The Netherlands
v.degraaff@utwente.nl

Anne van de Venis
Dept. of Computer Science
University of Twente
Enschede, The Netherlands
a.j.vandevenis@student.utwente.nl

Maurice van Keulen
Dept. of Computer Science
University of Twente
Enschede, The Netherlands
m.vankeulen@utwente.nl

Rolf A. de By
Fac. of Geo-Information Science
& Earth Observation (ITC)
University of Twente
Enschede, The Netherlands
r.a.deby@utwente.nl

ABSTRACT

Recommender systems have been around for decades to help people find the best matching item in a pre-defined item set. Knowledge-based recommender systems are used to match users based on information that links the two, but they often focus on a single, *specific* application, such as movies to watch or music to listen to. In this paper, we present our *Interest-Based Recommender System* (IBRS). This knowledge-based recommender system provides recommendations that are *generic* in three dimensions: IBRS is (1) domain-independent, (2) language-independent, and (3) independent of the used social medium. To match user interests with items, the first are derived from the user's social media profile, enriched with a deeper semantic embedding obtained from the generic knowledge base DBpedia. These interests are used to extract personalized recommendations from a tagged item set from any domain, in any language. We also present the results of a validation of IBRS by a test user group of 44 people using two item sets from separate domains: greeting cards and holiday homes.

Keywords

Recommender systems, knowledge-based, DBpedia, social media, domain-independent, language-independent

General Terms

Algorithms, Design, Experimentation

Categories and Subject Descriptors

H.4.2 [**Information Systems Applications**]: Types of Systems—*Decision support*

1. INTRODUCTION

The aim of a *recommender system* (RS) is to help people find the items they are most interested in. A requirement to provide *personalized* recommendations is that the RS has knowledge of the *person* using it. In 2013, Facebook claimed to have 1.11 billion active users [1], and the top-100 *pages* alone currently have a total of 5.87 billion *facebook-likes* [2]. The items that people express a preference for on social media, whether through a *like* of a Facebook page, a *follow* on Twitter, or a *tip* on the renewed FourSquare, can be taken to disclose personal traits of interest and the things they want to be associated with. This vast amount of information is the starting point for our *Interest-Based Recommender System* (IBRS).

But what people express their preference for on social media, cannot always directly be related to commonly used tags or words in descriptions in an existing item set. These items are often example instances of broader concepts. For example: *Cristiano Ronaldo* has 103 million *facebook-likes* at the time of writing, whereas Soccer (66 million) and Football (46 million) have considerably fewer facebook-likes.[1] Tag sets or descriptions, on the other hand, are more likely to contain these broader concepts, as for example is the case in greeting cards, sports equipment, or campsites with soccer fields. In fact, one of our validation item sets contains tagged greeting cards with practically only generic terms such as soccer/football. To bridge this generalization gap in a domain- and language-independent way, we use the multilingual, generic knowledge base DBpedia to automatically detect broader concepts. We call these concepts the user's *interests*. In this paper, we validate our hypothesis that automated user interest detection can also be used to select preferred items in an item set, independent of the item set domain, language and used social medium. As a boundary requirement to our solution, the cold-start problem, as for example discussed by Bobadilla et al. [3], needs to be circumvented. The system we propose shall be seen as a feature of a larger recommender system, either to bootstrap or to

[1] Synonyms like this one cause problems as well, and are discussed in more detail in Section 3

support that system, rather than as a stand-alone system.

In addition to the recommendation approach we propose in this paper, we also present the results of a validation thereof. A user group of 44 people tested our RS, using item sets from two completely different domains: greeting cards and holiday homes. Both the recommendation selection, as well as the explanation interface were validated by these users, using their own social media profile.

This paper is further structured as follows: related work is discussed in Section 2, the motivation behind this research is discussed in Section 3, the IBRS technology is presented in Section 4, while the validation approach and results are laid out in Section 5, and Section 6 finally contains concluding remarks and hints at future work.

2. RELATED WORK

The creation of a RS that makes use of social media or DBpedia is not a new ambition. Social media have especially received much attention in the field of content-based recommender systems. Fijałkowski and Zatoka presented an architecture of a recommender system for e-commerce based on Facebook profiles [4]. Guy et al. proposed five recommender types, based on social media and/or tags [5]. In their approach, they also presented the users with recommendation explanation. The social media they focus on however, are not of the mainstream type, but specific for the Lotus Connections suite. The system of He et al., on the other hand, uses common social media [6]. Whereas they claim to overcome the cold-start problem, their system appears to still suffer from the new item cold-start problem, as described by Bobadilla et al. [3].

The creation of a RS based on DBpedia has also received quite some attention already, especially in the field of music [7, 8] and movie [9, 10, 11, 12, 13] recommendation. Di Noia et al. took it a step further and also benefited from the integration of DBpedia in the *linked open data* (LOD) initiative. Their movie recommendations are not only based on DBpedia knowledge, but also on Freebase and LinkedMDB. A more generic approach to create a RS using LOD was done by Heitmann and Hayes [14], who use also use LOD to overcome the cold-start problem. Even though their validation is based on a music dataset, their approach has the genericity to be used for other applications as well. Our approach for broader concept detection through DBpedia is a form of knowledge-based query expansion. Liang et al. already showed in [15] that document recommendation based on the user's interests improves as a result of query expansion, or semantic-expansion as they call it.

What distinguishes our approach from other RS research, is that we use both social media profiles and DBpedia data to create a *generic* RS. Passant and Raimond, for example, created a RS based on exported social media profiles and DBpedia data in [8], but their approach is limited to the music-specific relations in DBpedia. To the best of our knowledge, the only other generic approach is TasteWeights by Bostandjiev et al. [16]. They build a user profile based on social media data, and then apply a collaborative filtering-based approach to select recommendations. This still implies all of the three cold-start problem categories: new item, new user, and new community, again as described by Bobadilla et al. [3]. As it is exactly our goal to overcome the cold-start problem, our approach is a hybrid between content-based and knowledge-based, according to the RS classification by Burke and Ramezani [17]. Basile, Lops et al. would classify our work as a top-down *semantics-aware* content-based RS [18, 19].

Our work is inspired by Shi et al.'s HeteRecom [20], which is based on the similarity calculation HeteSim [21]. Similar to their work, our ultimate goal is to find the matching paths between a user and the item set that carry the most weight. In this paper however, we focus on the detection of existing paths.

3. MOTIVATION

In this work, we aim to extract recommendations that are generic in three dimensions: the recommendation approach shall be independent of the item set domain, the item set language, and the used social medium. As a fourth criterium, it shall not suffer from any of Bobadilla's three cold-start problem categories. Below, we discuss the motivation for all of these challenges:

Domain-independence

As discussed in the previous section, currently most recommender systems based on knowledge bases and social media are focused on one specific domain. Independence of the item set domain only allows us to reuse the solution and its future improvements for multiple applications.

Language-independence

Similar to domain-independence as a requirement for reusability, a language-independent solution improves the RS's potential to be used in multiple applications. A sub-requirement of of language-independence is synonym-independence. As Zanardi and Capra pointed out in [22], synonyms are a typical RS problem, especially for tag-based RSs. The example of people facebook-liking either the *Soccer* page or the *Football* page from Section 1 already showed that people may facebook-like different pages, while referring to the same concept. Despite recent efforts by Facebook to merge pages about the same topic from different languages into one page, and improving the search functionality to help people finding such pages while searching for their name in a different language, still several pages exist to describe similar concepts.

Social medium-independence

From the first form of genericity, domain-independence, follows another requirement. Several social media, such as Facebook, LinkedIn, Twitter, Instagram, and Pinterest, are widely used, and each of these has its own focus. When one decides to create a RS for job vacancies, LinkedIn may be a more logical social medium to base the recommendations on than any of the other, while a RS for touristic hotspots will most likely lead to another choice. Therefore, to create a RS based on social media content that is domain-independent, it shall also be independent of the underlying social medium.

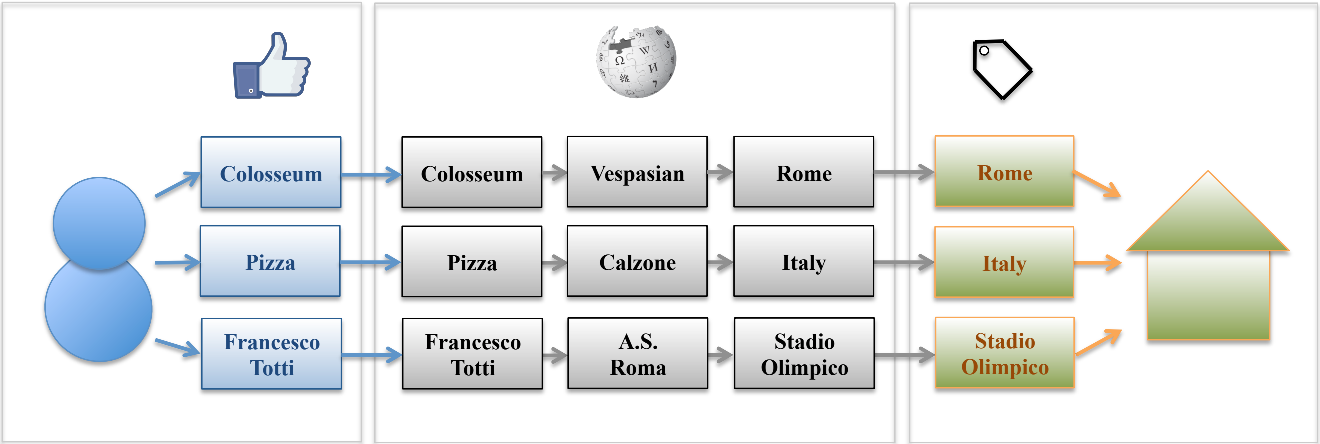


Figure 1: The IBRS concept, illustrated using the holiday home domain. A user's preferred items on social media are mapped onto knowledge base resources. Broader concepts are detected by exploring the knowledge base graph, and finally mapped onto tags in the item set database.

Cold-start problem

The cold-start problem has been widely discussed in RS literature. Bobadilla et al. categorized it into three subcategories: the new item problem, the new user problem, and the new community problem [3]. Knowledge-based RS have been designed to overcome all of these problems, but often require domain-specific knowledge.

Overcoming all of these four challenges at the same time has motivated us to create IBRS: a domain-independent, language-independent, social medium-independent, knowledge-based RS.

4. CONCEPT & TECHNOLOGY

The foundation of IBRS is the idea that people are more likely to be interested in items that have a not too distant relation with things we know they like. Although things people express a preference for on social media are typically in a different domain than our item set, they may still give hints towards a person's interests. In IBRS, we link the preferred items on social media to resources in the DBpedia *Resource Description Framework* (RDF) graph. We use this graph to explore related concepts, which are then matched with a known *tag set*, that is used to label the *item set*. As a final step, we rank the item set based on the number of *matched tags*. This concept is illustrated, using the holiday home domain, in Figure 1. In this example, the user facebook-liked the Colosseum, pizza, and Francesco Totti. These facebook-likes are mapped onto DBpedia, and the DBpedia RDF graph is explored to detect the broader concepts Rome, Italy, and Stadio Olimpico. These items are mapped onto holiday home tags, to ultimately match the user with a specific holiday home.

The remainder of this section is structured as follows: RDF graph exploration is discussed in Section 4.1. The data model of the IBRS abstraction layer is presented in Section 4.2. Section 4.3 presents a method for automated tag generation from descriptions. In Section 4.4 the ranking mechanism and Facebook-DBpedia mapping approach are presented. Section 4.5, finally, presents a short introduction of the IBRS prototype.

4.1 DBpedia graph exploration

After matching a facebook-like with a DBpedia resource, we traverse the RDF graph in exactly two steps. Since RDF tuples have a *subject*, *predicate* and *object*, RDF graphs are directed. Therefore, there are four possible different direction combinations to travel from node A through node B to its second neighbor C.[2] In Table 1, we show the top-10 of second neighbors when traversing the DBpedia graph starting from the *Eiffel Tower* as node A, using all four possible direction combinations. DBpedia pages in italics also occur as tags in at least one of our two validation sets, which are discussed in detail in Section 5. The first approach, $A \rightarrow B \rightarrow C$, leads to results describing France, influential French people, and several other buildings in France. The second approach, $A \leftarrow B \rightarrow C$, has some overlap with the first approach, but also contains several results unrelated to France, such as Los Angeles and the United States. The third approach, $A \leftarrow B \leftarrow C$, shows some remarkable buildings throughout Europe, but also very unrelated lists towards the bottom of the top-10. The fourth and final approach, $A \rightarrow B \leftarrow C$, results in several famous French people, especially scientists. Other starting points show similar results: the third approach, $A \leftarrow B \leftarrow C$, shows promising results for single domain recommendations, whereas the first approach shows the best results for broader concept detection. Since our aim is to match these second neighbors with a tag set, we use the first approach, $A \rightarrow B \rightarrow C$.

4.2 Abstraction layer data model

To ensure IBRS genericity, an abstraction layer is used on top of the underlying data source, such as a product database. This abstraction layer can consist of physical tables, views, or a mix thereof, but we will refer to its items as tables from here on. The abstraction layer contains two entity tables:

[2] Depending on the directions of the relationships, and the existence of bi-directional relationships, node A may be equal to node C, as can also be seen in Table 1.

Rank	$A \rightarrow B \rightarrow C$ (#)	$A \leftarrow B \rightarrow C$ (#)	$A \leftarrow B \leftarrow C$ (#)	$A \rightarrow B \leftarrow C$ (#)
1	*Paris* (20)	*Eiffel Tower* (41)	*Eiffel Tower* (7)	Paul Langevin (51)
2	*France* (20)	*France* (17)	Palácio de Ferro (3)	Léon Foucault (48)
3	*Eiffel Tower* (7)	*Paris* (15)	Cologne Cathedral (2)	Jean Témerson (48)
4	Manuel Valls (6)	*Los Angeles* (4)	Eiffel Bridge, Ungheni (2)	Frédéric Passy (45)
5	François Hollande (6)	British Library (4)	Souleuvre Viaduct (2)	L.A. de Bougainville* (45)
6	Unitary state (6)	Bonnétable (4)	Samuel Hibben (2)	Cecile de Brunhoff (45)
7	*French language* (6)	Aarhus University (4)	Casa de Fierro (2)	Adrien-Marie Legendre (45)
8	Anne Hidalgo (6)	Garabit viaduct (4)	Modern Marvels episodes* (2)	Robert Perrier (45)
9	Bonnétable (4)	St Paul's Cathedral (4)	Monopoly editions USA* (2)	Paul Lévy (math.)* (45)
10	Garabit viaduct (4)	*United States* (4)	Garabit viaduct (2)	Émile Drain (45)

Table 1: Top-10 of second neighbor nodes C through DBpedia graph exploration in multiple directions for the *Eiffel Tower* resource as node A. Numbers between brackets indicate number of paths between that node and the Eiffel Tower node. Items in italics also occur as tags in at least one of our two validation tag sets. Items marked with an asterisk are abbreviated.

ABSTRACT_ITEMS and TAGS, and one relationship table: ABSTRACT_ITEMS_TAGS, as depicted in Figure 2.

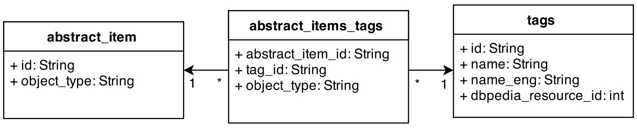


Figure 2: Abstraction layer data model

The ABSTRACT_ITEMS table contains the ID and OBJECT_TYPE of the items in the item set. The OBJECT_TYPE field allows us to use one IBRS instance for the recommendation of multiple item sets.

The TAGS table contains the tag's ID, NAME, and DBPEDIA_RESOURCE_ID. The NAME field can be used in the language of the item set tags. Since we have one item set that is tagged in Dutch, and one item set that is tagged in English, we added the NAME_ENG field for English tags. The DBPEDIA_RESOURCE_ID is cached in the database for better performance.

The ABSTRACT_ITEMS_TAGS table is a regular relation table containing the ABSTRACT_ITEM_ID and TAG_ID. It also contains the abstract item TYPE for improved join executions.

4.3 Tag generation

In case an item set is not tagged, but does contain descriptive texts, tags can be extracted automatically. Natural language processing algorithms can be used for this purpose, such as the named entity extraction and disambiguation approach by Habib et al. [23]. We used Habib's approach with a manually trained model to extract named entities from holiday home descriptions. A drawback of this approach is that descriptions are often the result of free-text input. Phrases such as "only a 3 hour flight from Amsterdam" or "25 kilometers from the border with France" led to correctly extracted named entities, but semantically not the best tags to distinguish this object from others. Therefore, we additionally removed those tags that tagged a holiday home with another country than the one it is located in. In total, this approach allowed us to assign 455,777 (non-unique) tags to 42,148 holiday homes, from which 106,430 tags (of which 12,151 unique) could be mapped onto a DBpedia resource.

4.4 Ranking

The IBRS ranking method consists of four steps: (1) retrieving preferred items from social media, (2) matching these items with DBpedia resources, (3) extracting abstracts from DBpedia, (4) ranking items based on matched tags. For performance reasons, several items are cached offline.

Obtaining preferred items from social media

To map social media items while remaining independent of the social medium, we must take into account that not all APIs are the same. Some social medium APIs allow developers to find out what a user's friends prefer, while others limit the developer to information about the logged in user. Therefore, when using the Facebook Graph API, we limited ourselves to the NAME and CATEGORY elements of each facebook-liked page.

Matching social media items with DBpedia resources

Facebook-likes are mapped onto DBpedia resources through their name. Those facebook-pages that mapped onto ambiguous terms in DBpedia were filtered out. To create a more complete mapping, we used the category element to postfix the name of those pages pages for which the category element was filled with "movie," "tv show," or "musician/band." In these cases, we also checked if a page exists with the additional suffix "_(movie)," "_(TV_series)," or "_(band)" respectively. This leads to the following SPARQL query:

```
PREFIX dbpont: <http://dbpedia.org/ontology/>
PREFIX dbpres: <http://dbpedia.org/resource/>
# We use the prefixed versions here for readability

SELECT ?uri ?label
WHERE {
 # Find exact match with category suffix
 { ?uri dbpont:wikiPageID [].
     FILTER(?uri = dbpres:The_Net_(movie)) }

 # Or exact match without category suffix
 UNION { ?uri dbpont:wikiPageID [].
     FILTER(?uri = dbpres:The_Net) }

  # Or the label version
 UNION {?uri rdfs:label "The_Net"@en.}
 # Check if page has redirect
 UNION { dbpres:The_Net_(movie)
     dbpont:wikiPageRedirects ?uri}
 UNION { dbpres:The_Net
     dbpont:wikiPageRedirects ?uri}

 ?uri rdfs:label ?label.
 ?uri dbpont:wikiPageID ?wikiPageid.
 FILTER (langMatches(lang(?label),"en")).

 # Filter out ambiguous terms
 FILTER NOT EXISTS { ?uri
   dbpont:wikiPageDisambiguates ?disambiguates } .

 # Filter out Wikipedia categories
 MINUS {?uri rdf:type skos:Concept}
}
LIMIT 1
```

Using this approach on a test set of 11,674 unique Facebook pages, obtained from the likes of 309 users, we were able to match 2,240 (19.2%) Facebook-pages with a DBpedia resource.

Extracting abstracts from DBpedia

For all matched DBpedia resources, the abstracts are retrieved from the SPARQL endpoint provided by DBpedia [24] using the following query:

```
PREFIX dbpont: <http://dbpedia.org/ontology/>
PREFIX dbpres: <http://dbpedia.org/resource/>

SELECT DISTINCT
 ?o3 (count(?o3) as ?count) ?abstract ?label

WHERE {
 # UNION concatenation of mapped FB pages
 {dbpres:Vienna ?p1 ?o2} UNION
 {dbpres:Recommender_system ?p1 ?o2} UNION
 {dbpres:Computer_science ?p1 ?o2}

 # Neighboring object has Wikipage
 ?o2 dbpont:wikiPageID ?o2id ;

 # Neighboring object has neighbor
    ?p2 ?o3 .

 # Second neighbor object has Wikipage
 ?o3 dbpont:wikiPageID ?o3id ;
    dbpont:abstract ?abstract ;
    rdfs:label ?label .

 # English is used as an example
 FILTER(langMatches(lang(?abstract), 'en')) .
 FILTER(langMatches(lang(?label), 'en')) .

 # Second neighbor object must not be a category
 MINUS {?o3 rdf:type skos:Concept}
}

# 'Only' the 1000 most important abstracts
ORDER BY DESC(?count)
LIMIT 1000
```

Ranking items based on matched tags

Each tag that (1) has a DBPEDIA_RESOURCE_ID and (2) is contained in at least one of the downloaded abstracts, is marked as a *matched tag*. The item set is then ranked on the basis of the number of matched tags. As a final step, those items that are too close to a higher ranked item, based on a pre-defined distance function, are removed from the ranking. This last step is added to ensure diversity among the recommended items. For the recommendation of geographic objects, as for example in a geo-social RS like the one discussed in [25], one can think of the Euclidean distance, but for more generic purposes the cosine similarity (as for example discussed in [22]) of the item's tags may be a good starting point.

4.5 Prototype

For demonstration and validation purposes, we have created a prototype of IBRS, using the Cake PHP platform. The prototype can be used with either one's own Facebook profile, or by manually combining several DBpedia resources. It can be accessed through `http://ibrs.ewi.utwente.nl`.

5. VALIDATION

To validate our ranking mechanism, as well as to determine the user perception of recommendations with explanations, we validated IBRS in a carefully designed user study with a test user group of 44 people. We used two product sets from different domains to demonstrate its domain-independence: greeting cards and holiday homes. The greeting card set contains Dutch tags, while the holiday homes did not contain any tags, but only descriptions. From the holiday homes, we used the English descriptions to extract (English) tags, to emphasize the potential to use IBRS in a language-independent way.

This section is further structured as follows: Section 5.1 describes the item set details. In Section 5.2, we present the approach taken to validate both our ranking mechanism and the recommendation explanation interface. Section 5.3 finally, discusses the validation results.

5.1 Item set details

The first item set contains greeting cards from the Dutch company *Kaartje2Go* ("Card2Go"). People search through a collection of cards electronically, which are distributed through regular (non-electronic) mail by Kaartje2Go in name of the client. To facilitate the search, users can search for tags that have been entered manually by the Kaartje2Go employees. These tags, which are mostly in Dutch, are inconsistent in their completeness: for example some of the soccer cards are also tagged using the names of popular Dutch soccer teams, but not all of them. Less popular teams are never mentioned as tags. The top-10 of the translated greeting card tags can be found in Table 2.

The second item set contains holiday homes from the holiday home portal *EuroCottage*. This item set did not contain tags, but a description in one, two or three languages (Dutch, English and/or German). We followed the approach discussed in Section 4.3 to extract mentions of geographic places from the English holiday home descriptions. The top-10 of resulting tags can be found in Table 3. The advantage of extracting geographic places is that these also often have Wikipedia pages, which makes them suitable for the requirement that the tags need to have a DBPEDIA_RESOURCE_ID. Many pages of the holiday home descriptions were in German, even though they were entered into the system by the

Tag	Frequency
Birthday	7,535
Party	4,200
Love	2,521
Girl	2,268
Boy	2,084
Infant	2,056
Photograph	1,793
Marriage	1,543
Cool	1,381
Animals	1,373

Table 2: Top-10 of (translated) manual greeting card tags with a DBpedia reference, ordered by the number of cards with this tag

holiday home owners as English descriptions. As a result thereof, many German words or phrases were extracted as geographical references, since the model was trained for English descriptions. However, the impact of these terms was practically zero, as these extracted tags were not matched with an English DBpedia resource.[3] For the validation, the holiday homes were plotted on a map that was zoomed in on Europe, since most holiday homes in the set are located there. A relatively small subset of homes outside Europe could therefore not be displayed on the map, and were removed from the validation set, just as those without a coordinate pair.

Tag	Frequency
Florence	760
Siena	656
Mediterranean Sea	634
Tuscany	537
Legoland	513
Venice	508
Sotkamo	448
Europe	440
Ardennes	421
Pisa	363

Table 3: Top-10 of extracted tags for holiday homes with a DBpedia reference, ordered by the number of holiday homes with this tag

5.2 Validation approach

Our test users were not aware of *what* they were testing, except for the information that they were testing a RS. Most test users do not have a background in computer science, and none of them were aware of how IBRS works. We asked our test users to validate our algorithm through a total of 30 questions, split up into three batches of 10. Once a question had been answered, users could not return to that question. The first two batches were intended to validate our ranking mechanism, the third batch was intended to determine the user perception of recommendations with explanations, as compared to recommendations without explanations.

For the first ten questions, users were asked to compare greeting cards using the interface of Figure 3. On one side of the screen, an item from the top-10 greeting cards according to IBRS was shown. On the other side, a card was shown that was not tagged with any of the matched tags. We called these recommendations *Inverted IBRS*. IBRS and Inverted IBRS were shown on the left or right side at random.

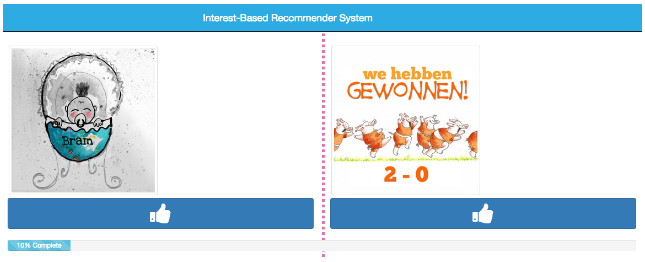
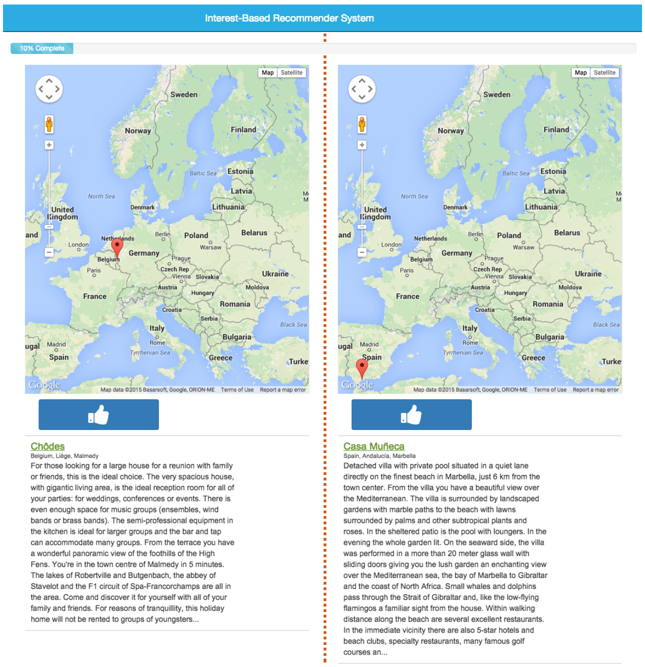


Figure 3: Validation interface for greeting card comparison

For the second batch of ten questions, our test users were presented with the choice between two holiday homes, in a similar way. Again, IBRS and Inverted IBRS were shown on the left or right side at random. For each holiday home, its location was shown on a map, with the name of the holiday home and the first 1000 characters of its description, as shown in Figure 4.



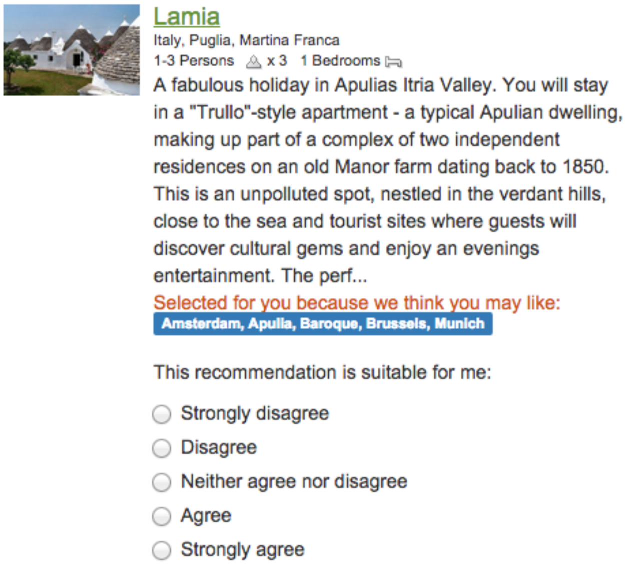
Chôdes
Belgium, Liège, Malmedy
For those looking for a large house for a reunion with family or friends, this is the ideal choice. The very spacious house, with gigantic living area, is the ideal reception room for all of your parties: for weddings, conferences or events. There is even enough space for music groups (ensembles, wind bands or brass bands). The semi-professional equipment in the kitchen is ideal for larger groups and the bar and tap can accommodate many groups. From the terrace you have a wonderful panoramic view of the foothills of the High Fens. You're in the town centre of Malmedy in 5 minutes. The lakes of Robertville and Butgenbach, the abbey of Stavelot and the F1 circuit of Spa-Francorchamps are all in the area. Come and discover it for yourself with all of your family and friends. For reasons of tranquillity, this holiday home will not be rented to groups of youngsters...

Casa Muñeca
Spain, Andalucía, Marbella
Detached villa with private pool situated in a quiet lane directly on the finest beach in Marbella, just 6 km from the town center. From the villa you have a beautiful view over the Mediterranean. The villa is surrounded by landscaped gardens with marble paths to the beach with lawns surrounded by palms and other subtropical plants and roses. In the sheltered patio is the pool with loungers. In the evening the whole garden lit. On the seaward side, the villa was performed in a more than 20 meter glass wall with sliding doors giving you the lush garden an enchanting view over the Mediterranean sea, the bay of Marbella to Gibraltar and the coast of North Africa. Small whales and dolphins pass through the Strait of Gibraltar and, like the low-flying flamingos a familiar sight from the house. Within walking distance along the beach are several excellent restaurants. In the immediate vicinity there are also 5-star hotels and beach clubs, specialty restaurants, many famous golf courses an...

Figure 4: Validation interface for holiday home comparison

The final batch of ten questions required the test users to rate a recommendation. Each of the holiday homes was one of the top-10 holiday homes according to IBRS. At random, a user was assigned to the group of users who received recommendations *with* an explanation, as shown in Figure 5, or *without* an explanation.

[3] Even though the approach can be applied to any language contained in the knowledge base, the tags are still matched with knowledge base resources in the tag language.



Lamia

Italy, Puglia, Martina Franca

1-3 Persons x 3 1 Bedrooms

A fabulous holiday in Apulias Itria Valley. You will stay in a "Trullo"-style apartment - a typical Apulian dwelling, making up part of a complex of two independent residences on an old Manor farm dating back to 1850. This is an unpolluted spot, nestled in the verdant hills, close to the sea and tourist sites where guests will discover cultural gems and enjoy an evenings entertainment. The perf...

Selected for you because we think you may like:

Amsterdam, Apulia, Baroque, Brussels, Munich

This recommendation is suitable for me:

○ Strongly disagree

○ Disagree

○ Neither agree nor disagree

○ Agree

○ Strongly agree

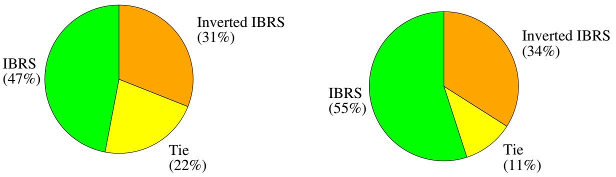
Figure 5: Cut-out of validation interface for holiday home recommendation rating. The lines in orange/blue contain the matched tags.

In test runs of the validation process, we determined that in a set-wise comparison of the two systems, users tended to prefer the set that was spread out over the map, rather than one that contained clusters of recommendations. Since Inverted IBRS is extremely spread out, due to the fact that items had no relation with the users or each other, this caused a bias in the validation results. Therefore, we decided to only compare the results item-wise. Furthermore, we removed tags with a negative connotation, such as "die," or "death."

5.3 Validation results

The first two batches of the validation were used to determine the potential of the IBRS ranking mechanism. The results are shown in the pie charts of Figure 6. Figure 6a shows which system was the test user's preferred system, based on a majority vote between the two systems. Most users participated in the validation of both the recommendation of greeting cards and holiday homes. Each batch was counted separately. 47% of the users preferred IBRS, 22% voted equally often for both of the systems, and 31% of the users preferred Inverted IBRS. In the pie chart of Figure 6b, the results are shown when the results of holiday homes with the greeting cards are combined per user. Since this increases the number of votes per user, ties are less common. In this scenario, 55% of the users preferred the IBRS results, while 34% preferred Inverted IBRS.

The final batch of the validation was used to determine the usefulness of the proposed recommendation explanation interface for holiday homes. The results of this batch are shown in the histograms of Figure 7. Contrary to our expectations, users preferred to receive recommendations without explanations. Using the 5-point Likert scale, the users who were presented with an interface with explanations rated the recommendations with an average score of 3.3772, while users without recommendation explanation rated the recommendations with a 3.4709 on average. From this validation, we can conclude that people that receive recommendations based on tags that do not describe them well, are more likely to reject a recommendation with a "strongly disagree," when they see the rationale behind the recommendation.

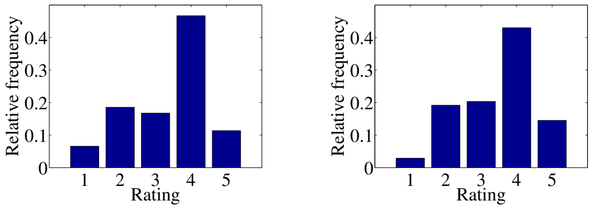


(a) Split out between greeting cards and holiday homes (batches counted separately)

(b) Overall (batches combined)

Figure 6: Most frequent choices per user for the first two batches of questions

Despite satisfying results with respect to the system's potential to rank recommendations for users, we should not forget that many aspects play a role in the decision-making that cannot (yet) be detected from Facebook profiles. When choosing either a greeting card, a holiday home, or anything else, one will always look at domain-specific item characteristics. For a greeting card, the user looks at colors, style, and the occasion the card is sent for. Similarly, for a holiday home, he looks at price, number of beds, the picture of the home, and the distance to the beach.



(a) With recommendation explanation; average rating: 3.3772.

(b) Without recommendation explanation; average rating: 3.4709.

Figure 7: Recommendation ratings split out by recommendation presentation interface

6. CONCLUSION

In this paper, we presented the approach behind IBRS. We discussed the concept of mapping items marked as preferred or liked in social media onto a generic knowledge-base, and query expansion using DBpedia. We presented the technology, including the abstraction layer, tag generation approach, and ranking mechanism. We also presented the validation results of a test user group. As said, we recommend to use the proposed and validated approach from this paper as a feature of a larger recommender system. In a more complete system, one also needs to take domain-specific features, as well as item popularity and other collaborative filtering features, into account. However, these features would contradict with our objective to create a generic RS that overcomes the cold-start problem, and therefore were not taken into account in this work.

Currently, IBRS uses all paths in the knowledge base graph as an indication for a useful recommendation. However, some paths in the graph actually form a reason *not* to recommend that item. For example, in the holiday home domain, a user is less likely to book a home in his own town, even though there may be many paths between him and that holiday home based on his local likes. Furthermore, some nodes are more useful than other for recommendation. DBpedia nodes like "European Central Time" have a lot of incoming paths, while it is unlikely that this actually forms an interest for this user. The next step for IBRS is to further improve the ranking mechanism by incorporating these characteristics and explore the possibility to automatically detect (negative) weights of paths.

7. ACKNOWLEDGEMENTS

This publication was supported by the Dutch national program COMMIT/. We also thank Mena Habib for his support in the tag generation process.

8. REFERENCES

[1] Facebook, "Facebook | photos." `https://www.facebook.com/facebook`, 2013.

[2] S. Bakers, "Statistics of the top facebook pages." `http://www.socialbakers.com/statistics/facebook/pages/total/`, 2013.

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

[4] D. Fijalkowski and R. Zatoka, "An architecture of a web recommender system using social network user profiles for e-commerce," in *Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on*, pp. 287–290, IEEE, 2011.

[5] I. Guy, N. Zwerdling, I. Ronen, D. Carmel, and E. Uziel, "Social media recommendation based on people and tags," in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pp. 194–201, ACM, 2010.

[6] J. He and W. W. Chu, *A social network-based recommender system (SNRS)*. Springer, 2010.

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

[8] A. Passant and Y. Raimond, "Combining social music and semantic web for music-related recommender systems," in *The 7th International Semantic Web Conference*, p. 19, Citeseer, 2008.

[9] R. Mirizzi, T. Di Noia, A. Ragone, V. C. Ostuni, and E. Di Sciascio, "Movie recommendation with DBpedia," in *IIR*, pp. 101–112, Citeseer, 2012.

[10] J. Golbeck and J. Hendler, "Filmtrust: Movie recommendations using trust in web-based social networks," in *Proceedings of the IEEE Consumer communications and networking conference*, vol. 96, pp. 282–286, University of Maryland, 2006.

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

[12] V. C. Ostuni, T. Di Noia, R. Mirizzi, D. Romito, and E. Di Sciascio, "Cinemappy: a context-aware mobile app for movie recommendations boosted by DBpedia," *SeRSy*, vol. 919, pp. 37–48, 2012.

[13] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos, "Moviexplain: a recommender system with explanations," in *Proceedings of the third ACM conference on Recommender systems*, pp. 317–320, ACM, 2009.

[14] B. Heitmann and C. Hayes, "Using linked data to build open, collaborative recommender systems.," in *AAAI spring symposium: linked data meets artificial intelligence*, pp. 76–81, 2010.

[15] T.-P. Liang, Y.-F. Yang, D.-N. Chen, and Y.-C. Ku, "A semantic-expansion approach to personalized knowledge recommendation," *Decision Support Systems*, vol. 45, no. 3, pp. 401–412, 2008.

[16] S. Bostandjiev, J. O'Donovan, and T. Höllerer, "TasteWeights: a visual interactive hybrid recommender system," in *Proceedings of the sixth ACM conference on Recommender systems*, pp. 35–42, ACM, 2012.

[17] R. Burke, "Hybrid web recommender systems," in *The adaptive web*, pp. 377–408, Springer, 2007.

[18] P. Lops, "Semantics-aware content-based recommender systems," 10 2014. Keynote at Workshop on New Trends in Content-based Recommender Systems.

[19] P. Basile, C. Musto, M. de Gemmis, P. Lops, F. Narducci, and G. Semeraro, "Content-based recommender systems + DBpedia knowledge = semantics-aware recommender systems," in *Semantic Web Evaluation Challenge*, pp. 163–169, Springer, 2014.

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

[21] C. Shi, X. Kong, Y. Huang, S. Y. Philip, and B. Wu, "HeteSim: A general framework for relevance measure in heterogeneous networks," *IEEE Transactions on Knowledge & Data Engineering*, no. 10, pp. 2479–2492, 2014.

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

[23] M. B. Habib and M. van Keulen, "Improving toponym disambiguation by iteratively enhancing certainty of extraction," in *Proceedings of the 4th International Conference on Knowledge Discovery and Information Retrieval, KDIR 2012, Barcelona, Spain*, (Spain), pp. 399–410, SciTePress, October 2012.

[24] DBpedia, "SPARQL explorer for http://dbpedia.org/sparql." `http://dbpedia.org/snorql/`, 2015.

[25] V. de Graaff, M. van Keulen, and R. A. de By, "Towards geosocial recommender systems," in *4th Intern. Workshop on Web Intelligence & Communities (WI&C 2012), Lyon, France*, ACM, 2012.

Extended Recommendation Framework: Generating the Text of a User Review as a Personalized Summary

Mickaël Poussevin
Sorbonne-Universités UPMC
LIP6 UMR 7606 CNRS
4 Place Jussieu, Paris, France
mickael.poussevin@lip6.fr

Vincent Guigue
Sorbonne-Universités UPMC
LIP6 UMR 7606 CNRS
4 Place Jussieu, Paris, France
vincent.guigue@lip6.fr

Patrick Gallinari
Sorbonne-Universités UPMC
LIP6 UMR 7606 CNRS
4 Place Jussieu, Paris, France
patrick.gallinari@lip6.fr

ABSTRACT

We propose to augment rating based recommender systems by providing the user with additional information which might help him in his choice or in the understanding of the recommendation. We consider here as a new task, the generation of personalized reviews associated to items. We use an extractive summary formulation for generating these reviews. We also show that the two information sources, ratings and items could be used both for estimating ratings and for generating summaries, leading to improved performance for each system compared to the use of a single source. Besides these two contributions, we show how a personalized polarity classifier can integrate the rating and textual aspects. Overall, the proposed system offers the user three personalized hints for a recommendation: rating, text and polarity. We evaluate these three components on two datasets using appropriate measures for each task.

1. INTRODUCTION

The emergence of the participative web has enabled users to easily give their sentiments on many different topics. This opinionated data flow thus grows rapidly and offers opportunities for several applications like e-reputation management or recommendation. Today many e-commerce websites present each item available on their platform with a description of its characteristics, average appreciation, ratings together with individual user reviews explaining their ratings.

Our focus here is on user - item recommendation. This is a multifaceted task where different information sources about users and items could be considered and different recommendation information could be provided to the user. Despite this diversity, the academic literature on recommender systems has focused only on a few specific tasks. The most popular one is certainly the prediction of user preferences given their past rating profile. These systems typically rely on collaborative filtering [9] to predict missing values in a *user/item/rating* matrix. In this perspective of rating prediction, some authors have make use of additional information sources available on typical e-commerce sites. [5] proposed to extract topics from consumer reviews in order to improve ratings predictions. Recently, [11] proposed to learn a latent space common to both textual reviews and product ratings, they showed that rating prediction was improved by such hybrid recommender systems. Concerning the information provided to the user, some models exploit review texts for ranking comments that users may like [1] or for answering specific user queries [17].

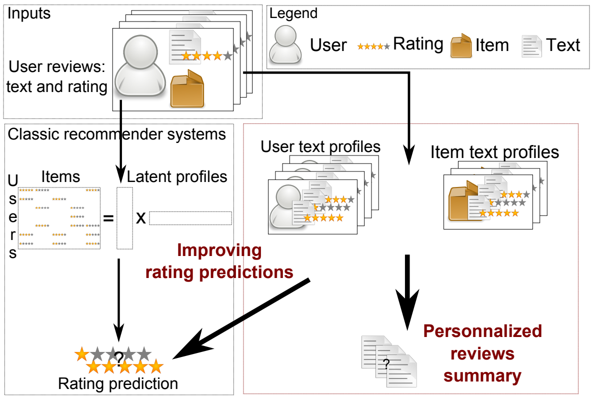


Figure 1: Our contribution is twofold: (1) improving rating predictions using textual information, (2) generating personalized reviews summaries to push recommender systems beyond rating predictions

We start here from the perspective of predicting user preference and argue that the exploitation of the information present in many e-commerce sites, allows us to go beyond simple rating prediction for presenting users with complementary information that may help him making his choice. We consider as an example the generation of a personalized review accompanying each item recommendation. Such a review is a source of complementary evidence for the user appreciation of a suggestion. Similarly as it is done for the ratings, we exploit past information and user similarity in order to generate these reviews. Since pure text generation is a very challenging task [2], we adopt an extractive summary perspective: the generated text accompanying each rating will be extracted from the reviews of selected users who share similar tastes and appreciations with the target user. Ratings and reviews being correlated, this aspect could also be exploited to improve the predictions. Our rating predictor will make use of user textual profiles extracted from their reviews and summary extraction in turn will use predicted ratings. Thus both types of information, predicted ratings and generated text reviews, are offered to the user and each prediction, rating and generated text, takes into account the two sources of information. Additional information could also be provided to the user. We show here as an example, that predicted ratings and review texts can be used to train a robust sentiment classifier which provides the user with a personalized polarity indication about the item. The modules of our system are evaluated on the two main tasks, rating prediction and summary extraction, and on the secondary task of sentiment prediction. For this, experiments are conducted on real datasets collected from *amazon.com* and *ratebeer.com* and models are compared to classical baselines.

The recommender system is compared to a classic collaborative filtering model using the mean squared error metric. We show that using both ratings and user textual profiles allows us to improve the performance of a baseline recommender. Gains are motivated from a more precise understanding of the key aspects and opinions included in the item and user textual profiles. For evaluating summary text generation associated to a couple (user, item), we have at our disposal a gold standard, the very review text written by this user on the item. Note that this is a rare situation in summary evaluation. However contrarily to collaborative filtering, there is no consensual baseline. We then compare our results to a random model and to oracle optimizing the ROUGE-n metric. They respectively provide a lower and an upper bound of the attainable performance. The sentiment classifier is classically evaluated using classification accuracy.

This article is organized as follows. The hybrid formulation, the review generator and the sentiment classifier are presented in section 2. Then, section 3 gives an extensive experimental evaluation of the framework. The overall gains associated to hybrid models are discussed in section 4. A review of related work is provided in section 5.

2. MODELS

In this section, after introducing the notations used throughout the paper, we will describe successively the three modules of our system. We start by considering the prediction of ratings [11]. Rating predictors answer the following question: *what rating will this user give to this item?* We present a simple and efficient way to introduce text profiles representing the writing style and taste of the user in a hybrid formulation. We then show how to exploit reviews and ratings in a new challenging task: *what text will this user write about this item?* We propose an extractive summary formulation of this task. We then proceed to describe how both ratings and text could be used together in a personalized sentiment classifier.

2.1 Notations

We use u (respectively i) to refer to everything relative to a user (respectively to an item) and the rating given by user u to the item i is denoted r_{ui}. U and I refer to anything relative to all users and all items, such as the rating matrix R_{UI}. Similarly, lower case letters are used for scalars or vectors and upper case letters for matrices. d_{ui} is the actual review text written by user u for item i. It is composed of κ_{ui} sentences: $d_{ui} = \{s_{uik}, 1 \leq k \leq \kappa_{ui}\}$. In this work, we consider documents as bags of sentences. To simplify notations, s_{uik} is replaced by s_{ui} when there is no ambiguity. Thus, user appreciations are quadruplets (u, i, r_{ui}, d_{ui}). Recommender systems use past information to compute a rating prediction $\hat{r}_{ui}$, the corresponding prediction function is denoted $f(u, i)$.

For the experiments, ratings and text reviews are split into training, validation and test sets respectively denoted S_{train}, S_{val} and S_{test} and containing m_{train}, m_{val} and m_{test} user appreciations (text and rating). We denote $S^{(u)}_{train}$, the subset of all reviews S_{train} that were written by user u and $m^{(u)}_{train}$ the number of such reviews. Similarly, $S^{(i)}_{train}$ and $m^{(i)}_{train}$ are used for the reviews on item i.

2.2 Hybrid recommender system with text profiles

Recommender systems classically use rating history to predict the rating $\hat{r}_{ui}$ that user u will give to item i. The hybrid system described here makes use of both *collaborative filtering* through matrix factorization and textual information to produce a rating as described in (1):

$$f(u,i) \quad = \quad \mu + \mu_u + \mu_i + \gamma_u . \gamma_i + g(u,i) \tag{1}$$

The first three predictors in equation (1) are biases (overall bias, user bias and item bias). The fourth predictor is a classical matrix factorization term. The novelty of our model comes from the fifth term (1) that takes into account text profiles to refine the prediction f. Our aim for the rating prediction is to minimize the following empirical loss function:

$$\underset{\mu,\mu_u,\mu_i,\gamma_u,\gamma_i,g}{\operatorname{argmin}} \; L \quad = \quad \frac{1}{m_{train}} \sum_{S_{train}} (r_{ui} - f(u,i))^2 \tag{2}$$

To simplify the learning procedure, we first optimize the parameters of the different components independently as described in the following subsections. Then we fine tune the combination of these components by learning weighting coefficients so as to maximize the performance criterion (2) on the validation set.

2.2.1 *Matrix factorization*

We first compute the different bias from eq. (1) as the averaged ratings over their respective domains (overall, user and item). For the matrix factorization term, we approximate the rating matrix R_{UI} using two latent factors: $R_{UI} \approx \Gamma_U \Gamma_I^t$. Both Γ_U and Γ_I are two matrices representing collections of latent profiles, with one profile per row. We denote γ_u (resp. γ_i) the row of Γ_U (resp. Γ_I) corresponding to the latent profile of user u (resp. item i).

The profiles are learned by minimizing, on the training set, the mean squared error between known ratings in matrix R_{UI} and the approximation provided by the factorization $\Gamma_U \Gamma_I^T$. This minimization problem described in equation (3), with an additional L2 constraint (4) on the factors is solved here using non-negative matrix factorization.

$$\Gamma_U^*, \Gamma_I^* = \underset{\Gamma_U,\Gamma_I}{\operatorname{argmin}} \qquad \| M_{train} \odot (R_{UI} - \Gamma_U \Gamma_I) \|_F^2 \tag{3}$$

$$+\lambda_U \|\Gamma_U\|_F^2 + \lambda_I \|\Gamma_I\|_F^2 \tag{4}$$

In this equation M_{train} is a mask that has the same dimensions as the rating matrix R_{UI}, an entry is 1 only if the corresponding review is in the training set and zero otherwise and $\odot$ is the element-wise product on matrices.

2.2.2 *Text profiles exploitation*

Let us denote π_u the profile of user u and $\sigma_t(\pi_{u'}, \pi_u)$ a similarity operator between user profiles. The last component of the predictor f in (1) is a weighted average of user ratings for item i, where weight $\sigma_t(\pi_{u'}, \pi_u)$ is the similarity between the text profiles $\pi_{u'}$ and π_u of users u' and u, the latter being the target user. This term takes into account the fact that two users with similar styles or using similar expressions in their appreciation of an item, should share close ratings on this item. The prediction term for the user/item couple (u, i) is then expressed as (5):

$$g(u,i) = \frac{1}{m^{(i)}_{train}} \sum_{S^{(i)}_{train}} r_{u'i} \sigma_t(\pi'_u, \pi_u) \tag{5}$$

Two different representations for the text profiles π_u of the users are investigated in this article: one is based on a latent representation of the texts obtained by a neural network autoencoder, the other relies on a robust bag of words coding. Each one is associated to a dedicated metric σ_t.

This leads to two formulations of g, and thus, to two rating prediction models. We denote the former f_A (autoencoder) and the latter f_T (bag of words). Details are provided below.

Bag of words.

A preprocessing step removes all words appearing in less than 10 documents. Then, the 100 000 most frequent words are kept. Although the number of features is large, the representation is sparse and scales well. π_u is simply the binary bag of words of all texts of user u. In this high dimensional space, the proximity in style between two users is well described by a cosine function, a high value indicates similar usage of words:

$$\sigma_t(\pi_{u'}, \pi_u) = \pi_{u'}\pi_u / (\|\pi_{u'}\|\|\pi_u\|) \tag{6}$$

Autoencoder.

The neural network autoencoder has two components: a coding operator and a decoding operator denoted respectively *cod* and *dec*. The two vectorial operators are learned so as to enable the reconstruction of the original text after a projection in the latent space. Namely, given a sentence s_{uik} represented as a binary bag of words vector, we obtain a latent profile $\pi_{s_{uik}} = cod(s_{uik})$ and then, we reconstruct an approximation of the sentence using $\hat{s}_{uik} = dec(\pi_{s_{uik}})$.

The autoencoder is optimized so as to minimize the reconstruction error over the training set:

$$cod^*, dec^* = \underset{cod,dec}{\operatorname{argmin}} \sum_{S_{train}} \frac{1}{\kappa_{ui}} \sum_{k=1}^{\kappa_{ui}} \|s_{uik} - dec(cod(s_{uik}))\|^2 \tag{7}$$

We use the settings proposed in [6]: our dictionary is obtained after stopwords removal and selecting the most frequent 5000 words. we did not use a larger dictionary such as the one used for the bag of word representation since it does not lead to improved performance and simply increases the computational load. All sentences are represented as binary bag of words using this dictionary. The coding dimension has been set to 1000 after a few evaluation trials. Note that the precise value of this latent space is not important and the performance is similar on a large range of dimension values. Both *cod* and *dec* use sigmoid units $\text{sig}(t) = \frac{1}{1+\exp(-t)}$:

$$\begin{aligned} \text{cod}(s_{uik}) &= \pi_{uik} = \text{sig}(W s_{uik} + b) \\ \text{dec}(\pi_{uik}) &= \text{sig}(W^T \pi_{uik} + b') \end{aligned} \tag{8}$$

Here, π_{uik} is a vector, W is a 5000x1000 weight matrix and sig() is a pointwise sigmoid operator operating on the vector $W s_{uik} + b$.

As motivated in [11, 5], such a latent representation helps exploiting term co-occurrences and thus introduces some semantic. It provides a robust text representation. The hidden activity of this neural network produces a continuous representation for each sentence accounting for the presence or absence of groups of words.

π_u is obtained by coding the vector corresponding to all text written by the user u in the past. It lies in a latent word space where a low Euclidean distance between users means a similar usage of words. Thus, for the similarity σ_t, we use an inverse Euclidean distance in the latent space:

$$\sigma_t(\pi_{u'}, \pi_u) = 1/(\alpha + \|\pi_{u'} - \pi_u\|) \tag{9}$$

2.2.3 *Global training criterion for ratings prediction*

In order to connect all the elementary components described above with respect to our recommendation task, we introduce weighting parameters β in (1). Thus, the initial optimization problem (2) becomes:

$$\begin{aligned} &\beta^* = \underset{\beta}{\operatorname{argmin}} \tfrac{1}{m_{train}} \textstyle\sum_{S_{train}} \\ &\Big(r_{ui} - \big(\beta_1\mu^* + \beta_2\mu^*_u + \beta_3\mu^*_i + \beta_4\gamma^*_u.\gamma^*_i + \beta_5 g(u,i)\big)\Big)^2 \end{aligned} \tag{10}$$

The linear combination is optimized using a validation set: this step guaranties that all components are combined in an optimal manner.

2.3 Text generation model

The goal here is to generate a review text for each (u,i) recommendation. During the recommendation process, this text is an additional information for users to consider. It should catch their interest and in principle be close to the one that user u could have written himself on item i. Each text is generated as an extractive summary, where the extracted sentences $s_{u'i}$ come from the reviews written by other users $(u' \neq u)$ about item i. Sentence selection is performed according to a criterion which combines a similarity between the sentence and the textual user profile and a similarity between the actual rating $r_{u'i}$ and the prediction made for (u,i), $\hat{r}_{ui}$ computed as described in section 2.2. The former measure could take into account several dimensions like vocabulary, sentiment expression and even style, here it is mainly the vocabulary which is exploited. The latter measures the proximity between user tastes. For the text measure, we make use of the σ_t similarity introduced in section 2.2. As before, we will consider two representations for texts (latent coding and raw bag of words). For the ratings similarity, we use $\sigma_r(r_{u'i}, r_{ui}) = 1/(1 + |r_{u'i} - r_{ui}|)$.

Suppose one wants to select a single sentence for the extracted summary. The sentence selection criterion will then be a simple average of the two similarities:

$$h(s_{u'i}, r_{u'i}, u', u, i) = \frac{\sigma_t(s_{u'i}, \pi_u) + \sigma_r(r_{u'i}, \hat{r}_{ui})}{2} \qquad (11)$$

Note that this function may score any piece of text. In the following, we then consider three possibilities for generating text reviews: The first one simply consists in selecting the best sentence $s_{u'i}$ among all the training sentences for item i with respect to h. We call it 1S for single sentence. The second one selects a whole review $d_{u'i}$ among all the reviews for i. The document is here considered as one long sentence. This is denoted CT for complete text. The third one is a greedy procedure that selects multiple sentences, it is denoted XS. It is initialized with 1S, and then sentences are selected under two criteria: relevance with respect to h and diversity with respect to the sentences already selected. Selection is stopped when the length of the text is greater than the average length of the texts of the target user. Algorithm 1 sums up the XS procedure for generating the text $\hat{d}_{ui}$ for the couple user u, item i.

```
Data: $u$, $i$, $S = \{(s_{u'i}, r_{u'i}\ u'\}$
Result: $\hat{d}_{ui}$
$s^*_{u'i} \leftarrow \underset{s_{u'i} \in S}{\text{argmax}}\big(h(s_{u'i}, r_{u'i}, u', u, i)\big)$;
$\hat{d}_{ui} \leftarrow s^*_{u'i}$;
Remove $s^*_{u'i}$ from $S$;
while length $\hat{d}_{ui}$ < averagelength($u$) do
    $s^*_{u'i} \leftarrow \underset{s_{u'i} \in S}{\text{argmax}}\big(h(s_{u'i}, r_{u'i}, u', u, i) - \cos(s_{u'i}, \hat{d}_{ui})\big)$ ;
    $\hat{d}_{ui} \leftarrow s^*_{u'i}$;
    Remove $s^*_{u'i}$ from $S$;
end
```

Algorithm 1: XS greedy procedure: selection of successive sentences to maximize both relevance and diversity. $\hat{d}_{ui}$ is the text that is generated, sentence after sentence.

2.4 Sentiment prediction model

We show here how polarity information about an item can be estimated by exploiting both the user predicted ratings and his textual profile. Exploiting both information sources improves the sentiment prediction performance compared with a usual text based sentiment classifier.

Polarity classification is the task of predicting whether a text d_{ui} (here of a review) is positive or negative. We use as ground truth the ratings r_{ui} and follow a standard thresholding procedure [15]: reviews rated 1 or 2 are considered as negative, while items rated 4 or 5 are positive. All texts that are rated 3 are ignored as it is unclear whether that are positive or negative: it strongly depends on the rating habits of the user.

For evaluation purpose, we consider two baselines. A first one only uses the rating prediction of our recommender system $f(u,i)$ as a label prediction, this value is then thresholded as indicated above. A second one is a classical text sentiment classifier. Denoting by d_{ui} the binary bag of word representation of a document and c_{ui} the binary label associated to the rating r_{ui}, one uses a linear SVM $s(d_{ui}) = d_{ui}.w$. Note that this is usually a strong baseline for the polarity classification task. Our final classifier will combine $f(u,i)$ and $s(d_{ui})$ in order to solve the following optimization problem:

$$w^* = \underset{w}{\text{argmin}} \sum_{S_{train}, r_{ui} \neq 3} \Big(1 - \big(d_{ui}.w + f(u,i)\big)c_{ui}\Big)_+ + \lambda \|w\|^2 \qquad (12)$$

with $(x)_+ = x$ when x positive and $(x)_+ = 0$ elsewhere. In the experimental section, we will also compare the results obtained with the two versions of our rating predictor: f_T and f_A (cf section 2.2.2).

Source	Subset names	#Users	#Items	#Reviews		
				#Training	#Validation	#Test
Ratebeer	RB_U50_I200	52	200	7200	900	906
	RB_U500_I2k	520	2000	388200	48525	48533
	RB_U5k_I20k	5200	20000	1887608	235951	235960
Amazon	A_U200_I120	213	122	984	123	130
	A_U2k_I1k	2135	1225	31528	3941	3946
	A_U20k_I12k	21353	12253	334256	41782	41791
	A_U210k_I120k	213536	122538	1580576	197572	197574

Table 1: Users, items & reviews counts for every datasets.

Subsets	μ	μ_u	μ_i	$\gamma_u.\gamma_i$	f_A	f_T
RB_U50_I200	0.7476	0.7291	0.3096	0.2832	**0.2772**	**0.2773**
RB_U500_I2k	0.6536	0.6074	0.3359	0.3168	**0.3051**	**0.3051**
RB_U5k_I20k	0.7559	0.6640	0.3912	0.3555	**0.3451**	**0.3451**
A_U200_I120	1.5348	2.0523	1.6563	1.7081	**1.4665**	1.4745
A_U2k_I1k	1.5316	1.4391	1.3116	1.0927	**1.0483**	**1.0485**
A_U20k_I12k	1.4711	1.4241	1.2849	1.0797	**1.0426**	**1.0426**
A_U210k_I120k	1.5072	2.1154	1.5318	1.2915	**1.1671**	**1.1678**

Table 2: Test performance (mean squared error) for recommendation. μ, μ_u, μ_i are the overall bias, user bias and item bias baselines. $\gamma_u.\gamma_i$ is the plain matrix factorization baseline. f_A, f_T are our hybrid recommender systems relying respectively on latent and raw text representations. The different datasets are described in table 1.

3. EXPERIMENTS

All three modules, ratings, text, sentiments, are evaluated independently since there is no global evaluation framework. These individual performances should however provide together a quantitative appreciation of the whole system.

We use two real world datasets of user reviews, collected from *amazon.com* [8] and *ratebeer.com* [11]. Their characteristics are presented in table 1.

Below, one presents first how datasets are preprocessed in 3.1. The benefits of incorporating the text in the ratings prediction for the recommender system are then discussed in section 3.2. The quality of the generated reviews is evaluated and analyzed in section 3.3 Finally, the performance of the sentiment classifier combining text and ratings is described in 3.4.

3.1 Data preprocessing

Reviews from different websites have different formats (rating scales, multiple ratings, . . .), they are then preprocessed to a unified format. Ratings are scaled to a 1 to 5 integer range. For titled reviews, the title is considered as the first sentence of the text of the review. Each dataset is randomly split into three parts: training, validation and test containing respectively 80%, 10% and 10% of the reviews.

As described in 2.2, two representations of the text are considered each with a different dictionary:

- for the autoencoder, we have selected the 5000 most frequent words, with a stopwords removal step; The autoencoder input vector is then a binary vector of dimension 5000.

- for the raw representation, we have selected the 100000 most frequent words appearing in more than 10 documents (including stopwords) and used a binary vector representation.

For the experiments, we consider several subsets of the databases with different numbers of users and items. Each dataset is built by extracting, for a given number of users and items, the most active users and the most commented items. Dataset characteristics are given in table 1.

Subsets	LL	μ_i	$\gamma_u.\gamma_i$	f_A	f_T	LL + f_A	LL + f_T
RB_U50_I200	5.35	5.12	6.01	5.57	5.57	**3.79**	**3.79**
RB_U500_I2k	7.18	10.67	9.73	8.55	8.55	**6.52**	6.92
RB_U5k_I20k	8.44	11.80	10.04	9.17	9.17	**8.33**	**8.35**
A_U200_I120	**10.00**	15.83	22.50	20.00	20.83	**10.00**	**10.00**
A_U2k_I1k	7.89	15.25	12.85	12.62	12.62	**7.54**	**7.54**
A_U20k_I12k	**6.34**	13.99	12.79	12.38	12.37	**6.29**	**6.29**
A_U210k_I120k	**6.25**	14.04	14.40	13.32	13.31	**6.22**	**6.22**

Table 3: Test performance (classification error) as polarity classifiers. LL stands for LibLinear (SVM), μ_i, $\gamma_u.\gamma_i$, f_A, f_T are the recommender systems as in table 2. LL + f_A and LL + f_T are two hybrid opinion classification models combining the SVM classifier and f_A and f_T recommender systems.

3.2 Recommender system evaluation

Let us first consider the evaluation of the rating prediction. The metric used here is the mean squared error (MSE) between rating predictions $\hat{r}_{ui}$ and actual ratings r_{ui}. The lower the MSE is, the better the model is able to estimate the correspondence between user tastes and items. Results are presented in table 2.

The models are referenced using the notations introduced in section 2.2. The first column corresponds to a trivial system which predicts μ the overall bias, the second predicts the user bias μ_u. Both give poor performance as expected.

The third column corresponds to the item bias μ_i baseline. It assumes that user taste is not relevant and that each item has its own intrinsic quality. The improvement with respect to μ and μ_u is important since MSE is halved. The fourth column corresponds to a nonnegative matrix factorization baseline, denoted $\gamma_u.\gamma_i$. It jointly computes latent representations for user tastes and items characteristics. Unsurprisingly, it is our best baseline.

It could be noted that performance tends to degrade when the subset size increases. This is a side effect associated to the review selection process used for building the different datasets. Smaller datasets contain the most active users and the most commented items. The estimation of their profiles benefits from the high number of reviews per user (and item) in this context.

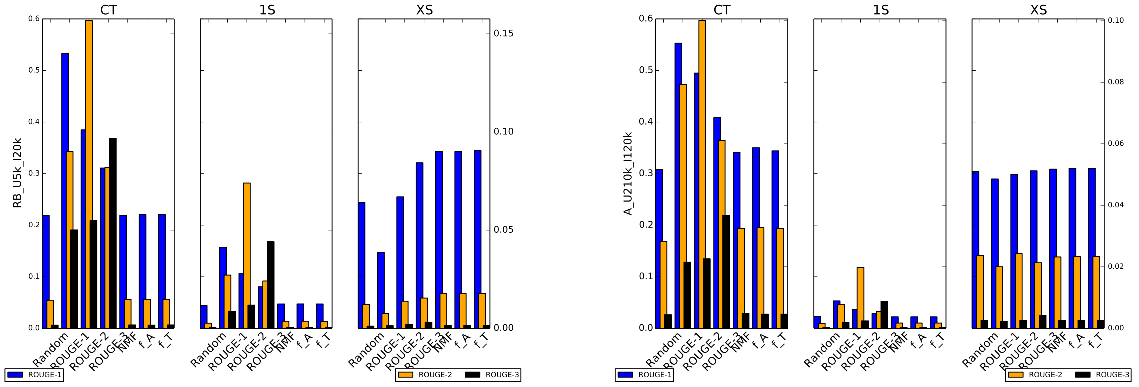
The last two columns refer to our hybrid recommender systems, using the two text representations introduced in section 2.2. Both f_A (autoencoder) and f_T (raw text) perform better than a baseline collaborative filtering system and both have similar approximation errors. The main difference between the systems comes from the complexity of the approach: during the learning step, f_T is much faster than f_A given the fact that no autoencoder optimization is required. On top of that, f_T remains faster in the inference step: the inherent sparsity of the bag of word representation enables f_T to provide faster computations than f_A. The autoencoder works in a smaller dimensional space but it is not sparse.

3.3 Text generation evaluation

We move on now to the evaluation of the personalized review text generation module. Since we are using an extractive summary procedure, we make use of a classical loss used for summarization systems: we use a recall-oriented ROUGE-n metrics, by comparing the generated text against the actual text of the review produced by the user. As far as we know, generating candidate reviews has never been dealt with in this context and this is a novel task. The ROUGE-n metric is the proportion of n-grams of the actual text found in the predicted (candidate) text, we use $n = \{1, 2, 3\}$. The higher ROUGE-n is, the better the quality of the candidate text is. A good ROUGE-1 means that topics or vocabulary are correctly caught while ROUGE-2 and ROUGE-3 are more representative of the user's style.

A first baseline is given by using a random scoring function h (instead of the formulation given in (11)). It provides a lower bound of the performance. Three oracles are then used to provide an upper bound on the performance. They directly optimize the metrics ROUGE-n from the data on the test set.A matrix factorization baseline is also used. It is a special case of our model where no text information is used. This model computes a similar score for all the sentences of a given user and relative to an item. When one sentence only is selected, it is taken at random among the sentences of this user for the item. With greedy selection, the first sentence is chosen at random and then the cosine diversity term (algorithm 1) allows a ranking of the next candidate sentences. Our proposed method is evaluated with the two different user profile π_u representation (auto-encoder and raw text). The performance of these seven models on the the two biggest datasets with respect to the three metrics are aggregated in figure 2.

An histogram corresponds to a text selection entity (whole review text, best single sentence, greedy sentence selection. Groups in the histograms (respectively row block of the tables) are composed of three cells corresponding respectively to the ROUGE-1, -2, -3 metrics. Not surprisingly, the results for the single sentence selection procedure (1S) are always worse than for the other two (CT: complete review and XS: multiple sentences). This is simply because a sentence contains fewer words than a full review and it can hardly share more n-grams than the full text with the reference text. For the *ratebeer.com* datasets, selecting a set of sentences clearly offers a better performance than selecting a whole review in all cases. Texts written to describe beers also describe the tasting experience. Was it in a bar or at home ? Was it a bottle or on tap ? Texts of the community share the same structure and vocabulary to describe both the tasting and the flavors of the beer. Most users write short and precise sentences. This is an appropriate context for our sentence scoring model, where the habits of users are caught by our recommender systems. The performance slightly decreases when the size of the dataset is increased. As before, this is in accordance with the selection procedure of these datasets which focuses first on the most active users and commented items. For Amazon, the conclusion is not so clear and depending on the conditions, either whole reviews or selected



(a) RateBeer experiments

(b) Amazon experiments

Figure 2: Histograms of the performances of the summarizer on the two biggest datasets. The scores of the ROUGE-1 metrics are represented in blue while the scores of the ROUGE-2 and ROUGE-3 metrics are in yellow and black. 7 models are compared: random, 3 oracles, NMF based model, f_A and f_T based models. 3 frameworks are investigated: CT (review extraction), 1S (One sentence extraction), XS (Multiple sentence extraction).

sentences get the best score. It is linked to the higher variety in the community of users on the website: well structured sentences like those present in RateBeer are here mixed here with different levels of English and troll reviews.

The different models, overall, are following a clear hierarchy. First, stating the obvious, the random model has the worst performance. Then, using a recommender system to select relevant sentences helps in terms of ROUGE-n performance. Using the text information brings most of the time only a small score improvement. Overall our models only offer small improvements here with respect to random or NMF text selection. After analyzing this behavior, we believe that this is due to the shortness of the text reviews, to their relatively standardized form (arguments are very similar from one review to another), to the peaked vocabulary distribution of the reviews, and to the nature of ROUGE. The latter is a classical recall oriented summarization evaluation measure, but does not distinguishes well between text candidates in this context. This also shows that there is room for improvement on this aspect.

Concerning the oracle several conclusions can be drawn. For both single sentence and complete text selection, the gap between the ROUGE measures and the proposed selection method is important suggesting that there is still room for improvements here too. For the greedy sentence selection, the gap between the oracles and the hybrid recommender systems is moderate suggesting that the procedure is here fully efficient. However this conclusion should be moderated. It can be observed that whereas, ROUGE is effectively an upper bound for single sentence or whole review selection, this is no more the case for multiple sentences selection. Because of the complexity of selecting the best subset of sentences according to a loss criterion (which amounts at a combinatorial selection problem) we have been using a suboptimal forward selection procedure: we first select the best ROUGE sentence, then the second best, etc. In this case the ROUGE procedure is no more optimal.

Concerning the measures, the performance decreases rapidly when we move from ROUGE-1 to ROUGE-2, 3. Given the problem formulation and the context of short product reviews, ROUGE-2,3 are clearly too constraining and the corresponding scores are not significant.

3.4 Sentiment classification evaluation

The performance of the different models, using the sentiment classification error as an evaluation metric, are presented in table 3. Because they give very poor performance, the bias recommendation models (μ and μ_u) are not presented here. The item bias μ_i, second column, gives a baseline, which is improved by the matrix factorization $\gamma_u.\gamma_i$, third column. Our hybrid models f_A, fourth column, and f_T, fifth column, have lower classification errors than all the other recommender systems. The first column, LL is the linear support vector machine (SVM) baseline. It has been learnt on the training set texts, and the regularization hyperparameter has been selected using the validation set. Our implementation relies on liblinear (LL) [4].

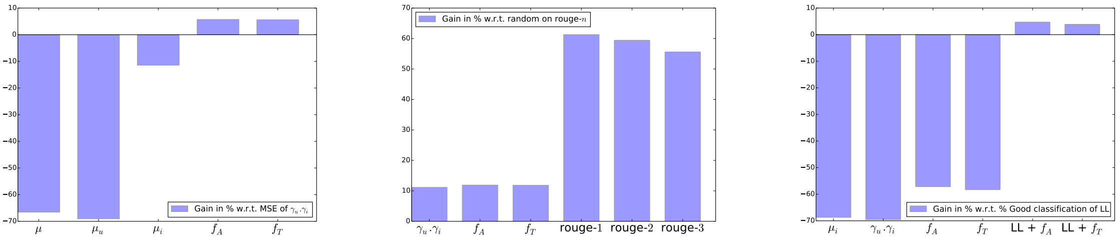
Its performance is better than the recommender systems but it should be noted that it makes use of the actual text d_{ui} of the review, whereas the recommender systems only use past information regarding user u and item i. Note that even in this context, the recommender performance on RateBeer is very close to the SVM baseline.

It is then possible to combine the two models, according to the formulation proposed in section 2.4. The resulting hybrid approaches, denoted LL + f_A and LL + f_T, exploit both text based decision (SVM) and user profile (f_A and f_T). This combined model shows good classification performance and overcomes the LL baseline in 4 out of 7 experiments in table 3, while performing similarly to LL in the other 3 experiments.

4. OVERALL GAINS

In order to get a global vision of the overall gain provided by the proposed approach, we summarize here the results obtained on the different tasks. For each task, the gain with respect to the (task dependent) baseline is computed and averaged (per task) over all datasets. The metric depends on the task. Results are presented in figure 3.

For the mean squared error metric (figure 3a) the matrix



(a) Recommender systems.
Baseline=matrix factorization

(b) Summarizers.
Baseline=random selection procedure

(c) Opinion classifiers.
Baseline=SVM

Figure 3: Aggregated gains on the 3 tasks w.r.t. classic baselines: our hybrid recommender systems are better overall.

factorization is used as baseline. The user bias μ_u heavily fails to generalize on two datasets. The item bias is closer to the baseline (-11.43%). Our hybrid models, which uses texts to refine user and item profiles bring a gain of 5.71% for f_A, 5.63% for f_T. This demonstrates the interest of including textual information in the recommender system. Autoencoder and raw text approaches offer similar gains, the latter approach being overall faster.

For the text generation, we take the random model as baseline and results are presented in figure 3b. The gain is computed for the three investigated framework (CT: review selection, 1S: one sentence selection, XS: multiple sentence selection) and per measure (ROUGE-1, 2, 3) and then averaged to one overall gain. ROUGE-n oracles clearly outperform other models, which seems intuitive. The different recommender systems have very close behaviors with respective gains of 11.15% (matrix factorization), 11.89% (auto-encoder), 11.83% (raw text). Here textual information helps but does not clearly dominate ratings and provide only a small improvement. Remember that although performance improvement with respect to baselines is desirable, the main novelty of the approach here is to propose a personalized summary generation together with the usual rating prediction.

For the opinion classifier, presented in figure 3c, the baseline consists in a linear SVM. Basic recommender systems perform poorly with respect to the baseline (LL). Surprisingly, the item bias μ_i (-68.71%) performs slightly better than matrix factorization $\gamma_u.\gamma_i$ (-69.54%) in the context of sentiment classification (no neutral reviews and binary ratings). Using textual information increases the performance. The autoencoder based model f_A (-57.17%) and raw text approach f_T (-58.31%) perform similarly. As discussed in 3.4, the linear SVM uses the text of the current reviews when the recommender systems does not. As a consequence, it is worth combining both predictions in order to exploit text and past profiles: the resulting models give respective gains of 4.72% (autoencoder) and 3.89% (raw text) w.r.t the SVM (LL).

5. RELATED WORK

Since the paper covers the topics of rating prediction, summarization and sentiment classification, we briefly present each of them.

5.1 Recommender systems

Three main families of recommendation algorithms have been developed [3]: content-based knowledge-based, and collaborative filtering. Given the focus of this work on consumer reviews, we considered collaborative filtering. For merchant websites the goal is to encourage users to buy new products and the problem is usually considered either as the prediction of a ranked list of relevant items for each user [13] or as the completion of missing ratings [9]. We have focused here on the latter approach for evaluation concerns: since we use data collected from third party sources.

5.2 Text summarization for consumer reviews

Early reference work [7] on consumer reviews has focused on global summarization of user reviews for each item. The motivation of this work was to extract the sentiments associated to a list of features from all the item review texts. The summarization took the form of a rating or of an appreciation of each feature. Here, contrarily to this line of work, the focus is on personalized item summaries for a target user. Given the difficulty of producing a comprehensive synthetic summary, we have turned this problem into a sentence or text selection process.

Evaluation of summaries is challenging: how to assess the quality of a summary when the ground truth is subjective? In our context, the review texts are available and we used them as the ground truth. We have used classical ROUGE-n summary evaluation measures [10].

5.3 Sentiment classification

Different text latent representations have been proposed in this scope: [14] proposed a generative model to represent jointly topic and sentiments and recently, several works have considered matrix factorization or neural network, in an attempts to develop robust sentiment recognition systems [6]. [16] go further and propose to learn two types of representation: a vectorial model is learned for word representation together with a latent transformation model, which allows the representation of negation and quantifiers associated to an expression.

We have investigated two kinds of representation for the texts: bag of words and a latent representation through the use of autoencoders as in [6]. [11] also uses a latent representation for representing reviews, although in a probabilistic setting instead in a deterministic one like we are doing here.

5.4 Hybrid approaches

In the field of recommendation, a first hybrid model was proposed by [5]: it is based on hand labeling of review sentences (topic and polarity) to identify relevant characteristics of the items. [11] pushes further the exploitation of texts, by using a joint latent representation for ratings and textual content with the objective of improving the rating accuracy. These two works are focused on rating prediction and do not consider delivering additional information to the user. Very recently, [19] has considered adding an explanation component to a recommender system. For that, they propose to extract some keywords from the review texts, which are supposed to explain why a user likes or dislikes an item. This is probably the work whose spirit is closest to ours but they do not provide a quantitative evaluation.

[7] combined opinion mining and text summarization on product reviews with the goal of extracting the qualities and defaults. [17] proposed a system for delivering personalized answers to user queries on specific products. They built the user profiles relying on topic modeling without any sentiment dimension. [1] proposed a personalized news recommendation algorithm evaluated on the Yahoo portal using user feedback, but it does not investigate ratings or summarization issues. Overall, we propose in this article to go beyond a generic summary of item characteristics by generating for each user a personalized summaries that is close to what they would have written about the item themselves.

For a long time, sentiment classification has ignored the user dimension and has focused for example on the conception of *"universal"* sentiment classifiers able to deal with a large variety of topics [15]. Considering the user has become an issue only very recently. [18] for example exploited explicit relations in social graphs for improving opinion classifiers, but their work is only focused on this aspect. [12] proposed to distinguish different rating behaviors and show that modeling the review authors in a scale ranging from connoisseur to expert offers a significant gain for an opinion prediction task.

In our work, we have experimented the benefits of considering the text of user reviews in recommender system for their performance as sentiment classifier. We have additionally proposed, as a secondary contribution, an original model mixing recommender systems and linear classification.

6. CONCLUSION

This article proposes an extended framework to the recommendation task. The general goal is to enrich classical recommender systems with several dimensions. As an example we show how to generate personalized reviews for each recommendation using extracted summaries. This is our main contribution. We also show how rating and text could be used to produce efficient personalized sentiment classifiers for each recommendation. Depending on the application, other additional information could be brought to the user. Besides producing additional information for the user, the different information sources can take benefit one from the other. We thus show how to effectively make use of text review and rating informations for building improved rating predictors and review summaries. As already mentioned, the sentiment classifiers also benefits from the two information sources. This part of the work demonstrates that multiple information sources could be useful for improving recommendation systems. This is particularly interesting since several sources are effectively available now at many online sites. Several new applications could be developed along the lines suggested here. From a modeling point of view, more sophisticated approaches can be developed. We are currently working on a multitask framework where the representations used in the different components are more closely correlated than in the present model.

7. REFERENCES

[1] D Agarwal, BC Chen, and B Pang. Personalized recommendation of user comments via factor models. *EMNLP'11*, 2011.

[2] M Amini and N Usunier. A contextual query expansion approach by term clustering for robust text summarization. *DUC'07*, 2007.

[3] R Burke. Hybrid recommender systems: Survey and experiments. *UMUAI'02*, 2002.

[4] R-E Fan, K-W Chang, C-J Hsieh, X-R Wang, and C-J Lin. Liblinear: A library for large linear classification. *JMLR'08*, 2008.

[5] G Ganu, N Elhadad, and A Marian. Beyond the Stars: Improving Rating Predictions using Review Text Content. *WebDB'09*, 2009.

[6] X Glorot, A Bordes, and Y Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML'11*, 2011.

[7] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. *KDD '04*, page 168, 2004.

[8] N Jindal and B Liu. Opinion spam and analysis. In *WSDM*, pages 219–230. ACM, 2008.

[9] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, pages 42–49, 2009.

[10] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *ACL Workshop: Text Summarization Branches Out*, 2004.

[11] J McAuley and J Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. *RecSys'13*, 2013.

[12] JJ McAuley and J Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. *WWW'13*, 2013.

[13] Matthew R McLaughlin and Jonathan L Herlocker. A Collaborative Filtering Algorithm and Evaluation Metric That Accurately Model the User Experience. In *SIGIR'04*, 2004.

[14] Q Mei, X Ling, M Wondra, H Su, and CX Zhai. Topic sentiment mixture: modeling facets and opinions in weblogs. In *WWW*. ACM, 2007.

[15] B Pang and L Lee. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2008.

[16] R Socher, B Huval, CD Manning, and A Ng. Semantic compositionality through recursive matrix-vector spaces. In *EMNLP'12*. ACL, 2012.

[17] C Tan, E Gabrilovich, and B Pang. To each his own: personalized content selection based on text comprehensibility. In *ICWDM'12*. ACM, 2012.

[18] C Tan, L Lee, J Tang, L Jiang, M Zhou, and P Li. User-level sentiment analysis incorporating social networks. In *KDD'11*. ACM, 2011.

[19] Y Zhang, G Lai, M Zhang, Y Zhang, Y Liu, and S Ma. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. 2014.

Automatic Selection of Linked Open Data features in Graph-based Recommender Systems

Cataldo Musto, Pierpaolo Basile, Marco de Gemmis,
Pasquale Lops, Giovanni Semeraro, Simone Rutigliano
Università degli Studi di Bari "Aldo Moro" - Italy
name.surname@uniba.it

ABSTRACT

In this paper we compare several techniques to automatically feed a graph-based recommender system with features extracted from the Linked Open Data (LOD) cloud. Specifically, we investigated whether the integration of LOD-based features can improve the effectiveness of a graph-based recommender system and to what extent the choice of the features selection technique can influence the behavior of the algorithm by *endogenously* inducing a higher *accuracy* or a higher *diversity*. The experimental evaluation showed a clear correlation between the choice of the feature selection technique and the ability of the algorithm to maximize a specific evaluation metric. Moreover, our algorithm fed with LOD-based features was able to overcome several state-of-the-art baselines: this confirmed the effectiveness of our approach and suggested to further investigate this research line.

Keywords

Recommender Systems, PageRank, Graphs, Linked Open Data, Feature Selection, Diversity

1. BACKGROUND

The Linked Open Data (LOD) cloud is a huge set of interconnected RDF statements covering many topical domains, ranging from government and geographical data to structured information about media (movies, books, etc.) and life sciences. The typical *entry point* to all this plethora of data is DBpedia [1], the RDF mapping of Wikipedia, which is commonly considered as the *nucleus* of the emerging *Web of Data*. Thanks to the wide-spread availability of this free machine-readable knowledge, a big effort is now spent to investigate whether and how the data gathered from the LOD cloud can be exploited to improve intelligent and adaptive applications, such a Recommender System (RS).

Recent attempts towards the exploitation of Linked Open Data to build RSs are due to Passant [6], who proposed a music recommender system based on semantic similarity calculations based on DBpedia properties. The use of DBpedia for similarity calculation is also the core of the work presented by Musto et al. in [4]: in this paper user preferences in music extracted from Facebook are used as input to find other relevant artists and to build a personalized music playlist. Recently, the use of LOD-based data sources has been the core of the ESWC 2014 Recommender Systems Challenge[1]: in that setting, the best-performing approaches [2] were based on ensembles of several widespread algorithms running on diverse sets of features gathered from the LOD cloud. However, none of the above described work tackles nor the issue of *automatically* selecting the best subset of LOD-based features, neither analyzes the impact of such selection techniques on different metrics as the *diversity* of the recommendations.

To this end, in this paper we propose a methodology to automatically feed a graph-based recommendation algorithm with features extracted from the LOD cloud. We focused our attention on graph-based approaches since they use a *uniform* formalism to represent both *collaborative* features (connections between users and items, expressed through ratings) and *LOD-based* ones (connections between different items, expressed through RDF statements). As graph-based algorithm we adopted PageRank with Priors [3]. Moreover, in this work we compared several techniques to automatically select the best subset of LOD-based features, with the aim to investigate to what extent the choice of the feature selection technique can influence the behavior of the algorithm and can endogenously lead to a higher *accuracy* or a higher *diversity* of the recommendations.

The rest of the paper is organized as follows: the description of our recommendation methodology is the core of Section 2, while the details of the experimental evaluation we carried out along with the discussion of the results are provided in Section 3. Finally, Section 4 sketches conclusions and future work.

2. METHODOLOGY

The main idea behind our graph-based model is to represent *users* and *items* as *nodes* in a graph. Formally, given a set of users $U = \{u_1 \dots u_n\}$ and a set of items $I = \{i_1 \dots u_m\}$, a graph $G = \langle V, E \rangle$ is instantiated. Given that for each user and for each item a node is created, $|V| = |U| + |I|$. Next, an edge connecting a user u_i with an item i_j is created for each positive feedback expressed by that user, so $E = \{(u_i, i_j) | likes(u_i, i_j) = true\}$.

CBRecSys 2015, September 20, 2015, Vienna, Austria.

[1] http://2014.eswc-conferences.org/important-dates/call-RecSys

Given this basic formulation, built on the ground of simple *collaborative*[2] data points, each item $i \in I$ can be provided with a relevance score. To calculate the relevance of each item, we used a well-known variant of the PageRank called *PageRank with Priors* [3]. Differently from PageRank, which assigns an evenly distributed prior probability to each node ($\frac{1}{N}$, where N is the number of nodes), *PageRank with Priors* adopts a non-uniform personalization vector assigning different weights to different nodes to get a bias towards some nodes (specifically, the preferences of a specific user). In our algorithm the probability was distributed by defining a simple heuristics, set after a rough tuning: 80% of the total weight is evenly distributed among items liked by the users (0% assigned to items disliked by the users), while 20% is evenly distributed among the remaining nodes. Damping factor was set equal to 0.85, as in [5].

Given this setting, the PageRank with Priors is executed for each user (this is mandatory, since the prior probabilities change according to user's feedbacks), and nodes are ranked according to their PageRank score which is in turn calculated on the ground of the connectivity in the graph. The output of the PageRank is a list of nodes ranked according to PageRank scores, labeled as L. Given L, recommendations are built by extracting from L only those nodes $i_1 \dots i_n \in I$.

2.1 Introducing LOD-based features

As stated above, our basic formulation does not take into account any data point different from users' ratings. The insight behind this work is to enrich the above described graph by introducing some *extra* nodes and some *extra* edges, defined on the ground of the information available in the LOD cloud. Formally, we want to define an extended graph $G_{LOD} = \langle V_{LOD-ALL}, E_{LOD-ALL}\rangle$, where $V_{LOD-ALL} = V \cup V_{LOD}$ and $E_{LOD-ALL} = E \cup E_{LOD}$. V_{LOD} and E_{LOD} represent the extra nodes and the extra edges instantiated by analyzing the data gathered from the LOD cloud, respectively.

As an example, if we consider the movie *The Matrix*, the property HTTP://DBPEDIA.ORG/PROPERTY/DIRECTOR encoding the information about the director of the movie is available in the LOD cloud. Consequently, an extra node *The Wachowski Brothers* is added in V_{LOD} and an extra edge, labeled with the name of the property, is instantiated in E_{LOD} to connect the movie with its director. Similarly, if we consider the property HTTP://DBPEDIA.ORG/PROPERTY/STARRING, new nodes and new edges are defined, in order to model the relationship between *The Matrix* and the main actors, as *Keanu Reeves*, for example. In turn, given that *Keanu Reeves* acted in several movies, many new edges are added in the graph and many new paths now connect different movies: these paths would not have been available if the only *collaborative data points* were instantiated.

It immediately emerges that, due to this novel enriched representation, the structure of the graph tremendously changes since many new nodes and many new edges are added to the model: the first goal of our experimental session will be to investigate whether graph-based RSs can benefit of the introduction of novel LOD-based features.

2.2 Selecting LOD-based features

Thanks to the data points available in the LOD cloud, many new information can be encoded in our graph. However, as the number of extra nodes and extra edges grows, the computational load of the PageRank with Priors grows as well, so it necessary to identify the subset of the most useful properties gathered from the LOD cloud and to investigate to what extent (if any) each of them improves the accuracy of our recommendation strategy.

A very *naive* approach may be to manually select the most relevant LOD-based features, according to simple heuristics or to domain knowledge (e.g. properties as *director*, *starring*, *composer* may be considered as relevant for the Movie domain, whereas properties as *runtime* or *country* may be not). This basic approach has several drawbacks, since it requires a manual effort, but it is also strictly domain-dependent.

To avoid this, we employed *features selection techniques* to automatically select the most promising LOD-based features. Formally, our idea is to take as input E_{LOD}, the overall set of LOD-based properties, and to produce as output $E_{LOD-FS_T} \subseteq E_{LOD}$, the set of properties a specific feature selection technique T returned as relevant. Clearly, the exploitation of a feature selection technique T also produces a set $V_{LOD-FS_T} \subseteq V_{LOD}$, containing all the LOD-based nodes connected to the properties in E_{LOD-FS_T}.

In this setting, given a FS technique T, PageRank will be executed against the graph $G_{LOD-T} = \langle V_{LOD-T}, E_{LOD-T}\rangle$, where $V_{LOD-T} = V \cup V_{LOD-FS_T}$ and $E_{LOD-T} = E \cup E_{LOD-FS_T}$. In the experimental session the effectiveness of seven different techniques for automatic selection of LOD-based features: *PageRank*, *Principal Component Analysis (PCA)*, *Support Vector Machines (SVM)*, *Chi-Squared Test (CHI)*, *Information Gain (IG)*, *Information Gain Ratio (GR)* and *Mininum Redundancy Maximum Relevance (MRMR)* [7]. Clearly, a complete description of these techniques is out of the scope of this paper. We will just limit to evaluate their impact on the overall *accuracy* and the overall *diversity* obtained by our algorithm.

3. EXPERIMENTAL EVALUATION

Our experiments were designed on the ground of four different research questions:

1. Do graph-based recommender systems benefit of the introduction of LOD-based features?

2. Do graph-based recommender systems exploiting LOD features benefit of the adoption of FS techniques?

3. Is there any correlation between the choice of the FS technique and the behavior of the algorithm?

4. How does our methodology perform with respect to state-of-the-art techniques?

Experimental design: experiments were performed by exploiting MovieLens[3] dataset, consisting of 100,000 ratings provided by 943 users on 1,682 movies. The dataset is positively balanced (55.17% of positive ratings) and shows an high sparsity (93.69%). Each user voted 84.83 items on average and each item was voted by 48.48 users, on average.

Experiments were performed by carrying out a 5-folds cross validation. Given that MovieLens preferences are expressed on a 5-point discrete scale, we decided to consider as *positive* ratings only those equal to 4 and 5. As recommendation algorithms we used the previously described PageRank

[2] We just modeled user-items couples, as in collaborative filtering algorithms

[3] http://grouplens.org/datasets/movielens/

with Priors, set as explained in Section 2. We compared the effectiveness of our graph-based recommendation methodology by considering three different graph topologies: G, modeling the basics *collaborative* information about user ratings; G_{LOD}, which enrichs G by introducing LOD-based features gathered from DBpedia, and G_{LOD-T} which lighten the load of PageRank with Priors by relying on the features selected by a FS technique T. In order to enrich the graph G, each item in the dataset was mapped to a DBpedia entry. In our experiments 1,600 MovieLens entries (95.06% of the movies) were successfully mapped to a DBpedia node. The items for which a DBpedia entry was not found were only represented by using *collaborative* data points. Overall, MovieLens entries were described through 60 different DBpedia properties. As feature selection techniques all the approaches previously mentioned were employed, while for the parameter K (the number of LOD-based features) three different values were compared: 10, 30 and 50. The performance of each graph topology was evaluated in terms of *F1-measure*. Moreover, we also calculated the overall running time[4] of each experiment. To answer the third research questions we also evaluated the *diversity* of the recommendations, calculated by exploiting the classical Intra-List Diversity (ILD). Statistical significance was assessed by exploiting Wilcoxon and Friedman tests.

Discussion of the Results: in the first experiment we evaluated the introduction of LOD-based features in graph-based recommender systems. Results are depicted in the first two columns of Table 1. As regards MovieLens, a statistically significant improvement ($p << 0.0001$, assessed through a Wilcoxon test) was obtained for all the metrics. As expected, the expansion of the graph caused an exponential growth of the run time of the algorithm. This is due to the fact that the expansion stage introduced many new nodes and many new edges in the graph (see Table 1). The growth is particularly significant since 50,000 new nodes and 78,000 new edges were added to the graph.

Next, we evaluated the impact of all the previously presented feature selection techniques in such recommendation setting. By analyzing the results provided in Table 2, it emerged that our graph-based recommendation strategy does not often benefit of the application of FS techniques. Indeed, when a very small number of properties is chosen (*K=10*), none of the configurations is able to overcome the baseline. By slightly increasing the value of parameter K (*K=30*), only three out of seven techniques (PageRank, PCA and mRMR) improve the F1-measure. Next, when more data points are introduced (with *K=50*) better results are obtained and the F1-measure of the baseline is always overcame. Given that the overall number of LOD-based properties was equal to 60, it is possible to state that most of the properties encoded in the extended graph G_{LOD} can be considered as relevant. Clearly, this is a very domain-specific outcome, which needs to be confirmed by more thorough analysis on different datasets. However, it is possible to state that the adoption of FS techniques requires a complete analysis of the usefulness of each of the properties encoded in the LOD. Overall, the best performing configuration was PCA, which was the only technique always overcoming the baseline with $K = 50$. A Friedman test also showed that PCA statistically overcomes the other techniques for all the metrics. Another interesting outcome which follows the use of FS techniques is the saving of computational resources to run PageRank with Priors on graph $G_{LOD-PCA}$. As shown in Table 1, the adoption of FS caused a huge decrease of the run time of the algorithms equal to 33.9% for MovieLens (from 880 to 581 minutes). This is due to the smaller number of information which are modeled in the graphs (-8.6% nodes and -4.8% edges).

	MovieLens		
	G	G_{LOD}	$G_{LOD+PCA}$
F1@5	0.5406	**0.5424**	**0.5424(*)**
F1@10	0.6068	**0.6083**	**0.6088(*)**
F1@15	0.5956	**0.5963**	**0.5970(*)**
F1@20	0.5678	**0.5686**	**0.5689(*)**
Run (min.)	72	880	581
K (LOD prop.)	0	60	50
Nodes	2,625	53,794	49,158
Edges	100,000	178,020	169,405

Table 1: Overall comparison among the baseline, the complete LOD-based graph and the LOD-based graph boosted by PCA. The configurations overcoming the baseline were highlighted in bold. The best-performing configuration is further highlighted with (*)

In *Experiment 3* we shifted the attention from F1-measure to different evaluation metrics, and we investigate whether the adoption of a specific FS technique can *endogenously* induce a higher diversity at the expense of a little F1. Results of the experiments are provided in Figure 1a. Due to space reasons, only the results for F1@10 are provided. In both charts we used four different symbols to identify the different behaviors of each technique. It emerged that CHI was the less useful technique, since it did not provide any significant benefit to neither F1 nor diversity. Next, PageRank provided a (small) improvement on F1 and it did not significantly change the diversity of the recommendations. It is noteworth that the larger increase in accuracy of PCA is balanced by a decrease in terms of diversity. On the other side, Gain Ratio obtained the overall best diversity of the recommendation but it decreases the F1 of the algorithm. To sum up, these results show that the choice of a particular FS technique has a significant impact on the overall behavior of the recommendation algorithm. As shown in the experiment, some techniques have the ability of inducing a higher diversity (or F1) at the expense of a little of F1 (or diversity, respectively), wheres other can provide a good compromise between both metrics. Clearly, more investigation is needed to deeply analyze the behavior of each technique, but these results already give some general guidelines which can drive the choice of the FS technique which best fits the requirements of a specific recommendation scenario.

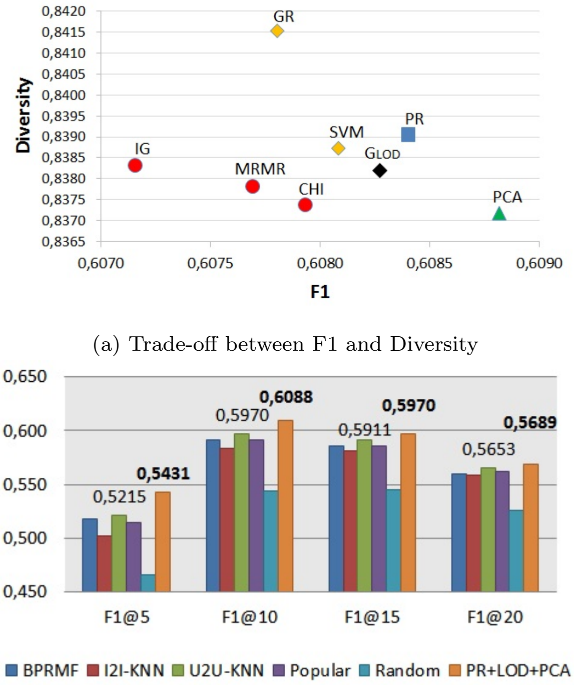
Finally, we compared the effectiveness of our methodology to the current state of the art. As baselines User-to-User CF (U2U-KNN), Item-to-Item CF (I2I-KNN), a simple popularity-based approach, a random baseline and the Bayesian Personalized Ranking Matrix Factorization (BPRMF) were used. We adopted the implementations available in MyMediaLite Recommender System library[5]. As regards

[4] Experiments were run on an Intel-i7-3770 CPU3.40 gHZ, with 32GB RAM.

[5] http://www.mymedialite.net/

MovieLens	#feat.	PR	PCA	SVM	CHI	IG	GR	mRMR
F1@5	10	0.5418	0.5406	0.5382	0.5414	0.5397	0.5372	0.5397
$G_{LOD} = 0.5424$	30	**0.5429(∗)**	0.5413	0.5413	0.5419	0.5396	0.5398	**0.5429(∗)**
	50	0.5412	**0.5431(∗)(↑)**	0.5421(∗)	0.5420(∗)	0.5412(∗)	0.5406(∗)	0.5421
F1@10	10	0.6069	0.6045	0.6043	0.6056	0.6039	0.6033	0.6039
$G_{LOD} = 0.6083$	30	**0.6084(∗)**	0.6081	0.6074	0.6070	0.6055	0.6059	0.6072(∗)
	50	0.6070	**0.6088(∗)(↑)**	0.6081(∗)	0.6079(∗)	0.6072(∗)	0.6078(∗)	0.6077
F1@15	10	**0.5964**	0.5948	0.5943	0.5955	0.5950	0.5938	0.5950
$G_{LOD} = 0.5963$	30	**0.5967(∗)**	**0.5967**	**0.5964**	**0.5967**	0.5955	0.5960	0.5961
	50	0.5955	**0.5970(∗)(↑)**	**0.5966(∗)**	**0.5972(∗)**	0.5962(∗)	**0.5968(∗)**	0.5962(∗)
F1@20	10	0.5684(∗)	0.5667	0.5666	0.5672	0.5668	0.5666	0.5668
$G_{LOD} = 0.5686$	30	0.5684	**0.5688**	0.5679	0.5679	0.5675	0.5675	0.5679
	50	0.5682	**0.5689(∗)(↑)**	0.5683(∗)	**0.5686(∗)**	0.5685(∗)	**0.5687(∗)**	0.5685(∗)

Table 2: Experiment 2. The configurations overcoming the baseline G_{LOD} are emphasized in bold. Next, for each technique, the number of features which led to the highest F1 is indicated with (∗). The overall highest F1 score for each metric is highlighted with (∗)(↑). The column of the feature selection technique which performed the best on a specific dataset is coloured in grey.



(a) Trade-off between F1 and Diversity

(b) Comparisons to baselines

Figure 1: Results of the experiments

U2U and I2I, neighborhood size was set to 80, while BPRMF was run by setting the factor parameter equal to 100. Results are depicted in Figure 1b. As shown in the plots, our graph-based RS outperforms all the baselines for all the metrics taken into account. It is worth to note that our approach obtained a higher F1 also when compared to a well-perfoming matrix factorization algorithm as BPRMF, thus this definitely confirmed the effectiveness of our approach.

4. CONCLUSIONS AND FUTURE WORK

In this work we proposed a graph-based recommendation methodology based on PageRank with Priors, and we evaluated different techniques to automatically feed such a representation with features extracted from the LOD cloud. Results showed that graph-based RSs can benefit of the infusion of novel knowledge coming from the LOD cloud and that a clear correlation between the adoption of a specific FS technique with the overall results of the recommender exitss, since some techniques *endogenously* showed the ability of increasing also the diversity of the recommendations generated by the algorithm. We also showed that our methodology was able to overcome several state-of-the-art baselines on both datasets. As future work, we will validate the approach by evaluating it on different dataset, and we will investigate the impact of LOD-based features with different learning approaches as Random Forest or SVM.

5. REFERENCES

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. *DBpedia: A nucleus for a Web of Open Data*. Springer, 2007.

[2] P. Basile, C. Musto, M. de Gemmis, P. Lops, F. Narducci, and G. Semeraro. Aggregation strategies for linked open data-enabled recommender systems. In *European Semantic Web Conference*, 2014.

[3] T. H. Haveliwala. Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search. *IEEE Trans. Knowl. Data Eng.*, 15(4):784–796, 2003.

[4] C. Musto, G. Semeraro, P. Lops, M. de Gemmis, and F. Narducci. Leveraging social media sources to generate personalized music playlists. In *EC-Web 2012*, volume 123 of *LNBIP*, pages 112–123. Springer, 2012.

[5] L. Page, S. Brin, R. Motwani, and T. Winograd. The pageRank citation ranking: bringing order to the web. 1999.

[6] A. Passant. dbrec - Music Recommendations Using DBpedia. In *International Semantic Web Conference, Revised Papers*, volume 6497 of *LNCS*, pages 209–224. Springer, 2010.

[7] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8):1226–1238, 2005.

Cross-Document Search Engine For Book Recommendation

Chahinez Benkoussas
Aix-Marseille Université, CNRS, LSIS UMR 7296
13397, Marseille. France
chahinez.benkoussas@lsis.org
Aix-Marseille Université, CNRS, CLEO OpenEdition UMS 3287, 13451
13397, Marseille. France
chahinez.benkoussas@openedition.org

Patrice Bellot
Aix-Marseille Université, CNRS, LSIS UMR 7296
13397, Marseille. France
patrice.bellot@lsis.org
Aix-Marseille Université, CNRS, CLEO OpenEdition UMS 3287, 13451
13397, Marseille. France
patrice.bellot@openedition.org

ABSTRACT

A new combination of multiple Information Retrieval approaches are proposed for book recommendation based on complex users' queries. We used different theoretical retrieval models: probabilistic as InL2 (Divergence From Randomness model) and language models and tested their interpolated combination. We considered the application of a graph based algorithm in a new retrieval approach to related document network comprised of social links. We called Directed Graph of Documents (DGD) a network constructed with documents and social information provided from each one of them. Specifically, this work tackles the problem of book recommendation in the context of CLEF Labs precisely Social Book Search track. We established a specific strategy for queries searching after separating query set into two genres "*Analogue*" and "*Non-Analogue*" after analyzing users' needs. Series of reranking experiments demonstrate that combining retrieval models and exploiting linked documents for retrieving yield significant improvements in terms of standard ranked retrieval metrics. These results extend the applicability of link analysis algorithms to different environments.

Keywords

Document retrieval, InL2, language model, book recommendation, PageRank, graph modeling, Social Book Search.

1. INTRODUCTION

There has been much work both in the industry and academia on developing new approaches to improve the performance of retrieval and recommendation systems over the last decade. The aim is to help users to deal with information overload and provide recommendation for books, restaurants or movies. Some vendors have incorporated recommendation capabilities into their commerce services, such as Amazon.

Existing document retrieval approaches need to be improved to satisfy users' information needs. Most systems use classic information retrieval models, such as language models or probabilistic models. Language models have been applied with a high degree of success in information retrieval applications [29–31]. This was first introduced by Ponte and Croft in [27]. They proposed a method to score documents, called *query likelihood* in two steps: estimate a language model for each document and then rank documents according to the likelihood scores resulting from the estimated language model. Markov Random Field model, proposed by Metzler and Croft in [19] considers query term proximity in documents by estimating term dependencies in the context of language modeling approach. Alternatively, Divergence From Randomness model, proposed by Amati and Van Rijsbergen [2], measures the global informativeness of the term in the document collection. It is based on the idea :"*The more the term occurrences diverge from random throughout the collection, the more informative the term is*" [28]. One limit of such models is that the distance between query terms in documents is not considered.

Users' queries differ by their type of needs. In book recommendation, we identified two genres of queries : "*Analogue*" and "*Non-Analogue*" that we describe in the following sections. In this paper, the first proposed approach combines probabilistic and language models to improve the retrieval performances and show that the two models act much better

CBRecSys 2015, September 20, 2015, Vienna, Austria.

in the context of book recommendation.

In recent years, an important innovation in information retrieval is the exploitation of relationships between documents, e.g. Google's PageRank [25]. It has been successful in Web environments, where the relationships are provided by hyperlinks between documents. We present a new approach for linking documents to construct a graph structure that is used in retrieving process. In this approach, we exploit the PageRank algorithm for ranking documents with respect to users' queries. In the absence of manually-created hyperlinks, we use social information to create a Directed Graph of Documents (DGD) and argue that it can be treated in the same manner as hyperlink graphs. Our experiments will show that incorporating graph analysis algorithms in document retrieval improves the performance in term of the standard ranked retrieval metrics.

Our work focuses on search in the book recommendation domain, in the context of CLEF Labs Social Book Search track. We tested our approaches on collection contains Amazon/LibraryThing book descriptions and set of queries, called topics, extracted from the LibraryThing discussion forums.

2. RELATED WORK

This work is first related to the area of document retrieval models, more specially language models and probabilistic models. The unigram language models are most often used for ad hoc Information Retrieval work but several researchers explored the use of language modeling for capturing higher order dependencies between terms. Bouchard and Nie in [8] showed significant improvements in retrieval effectiveness with a new statistical language model for the query based on completing the query by terms in the user's domain of interest, reordering the retrieval results or expanding the query using lexical relations extracted from the user's domain of interest.

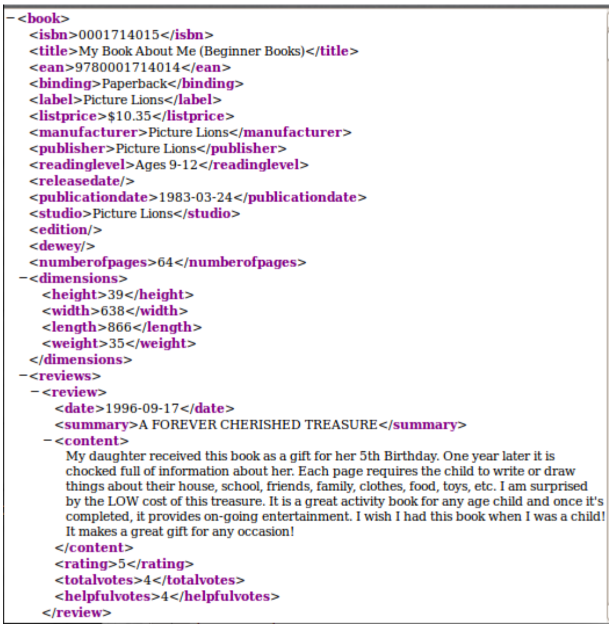
Divergence From Randomness (DFR) is one of several probabilistic models that we have used in our work. Abolhassani and Fuhr have investigated several possibilities for applying Amati's DFR model [2] for content-only search in XML documents. [1].

There has been an increasing use of techniques based on graphs constructed by implicit relationships between documents. Kurland and Lee performed structural reranking based on centrality measures in graph of documents which has been generated using relationships between documents based on language models [14]. In [16], Lin demonstrates the possibility to exploit document networks defined by automatically-generated content-similarity links for document retrieval in the absence of explicit hyperlinks. He integrates the PageRank scores with standard retrieval score and shows a significant improvement in ranked retrieval performance. His work was focused on search in the biomedical domain, in the context of PubMed search engine. Perhaps the main contrast with our work is that links were not induced by generation probabilities or linguistic items.

3. INEX SOCIAL BOOK SEARCH TRACK AND TEST COLLECTION

Social Book Search (SBS) task[1] aims to evaluate the value of professional and user's metadata for book search on the Web. The main goal is to exploit search techniques to deal with complex information needs and complex information sources that include user profiles, personal catalogs, and book descriptions.

The SBS task provides a collection of 2.8 million book description crawled by the University of Duisburg-Essen from Amazon[2] [4] and enriched with content from LibraryThing[3], which is an online service to help people catalog their books easly. Books are stored in XML files and identified by an ISBN. They contains information like: title information, Dewey Decimal Classification (DDC) code (for 61% of the books), category, Amazon product description, etc. Amazon records contain also social information generated by users like: tags, reviews, ratings (see Figure 1. For each book, Amazon suggests a set of "Similar Products" which represents a result of computed similarity based on content information and user behavior (purchases, likes, reviews, etc.) [13].



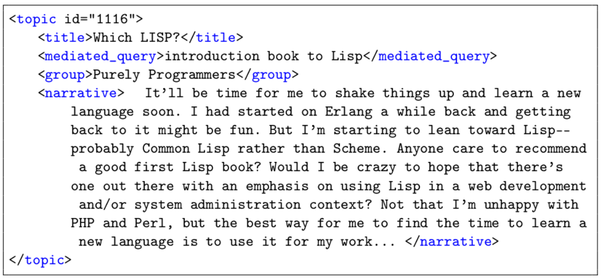
```
-<book>
   <isbn>0001714015</isbn>
   <title>My Book About Me (Beginner Books)</title>
   <ean>9780001714014</ean>
   <binding>Paperback</binding>
   <label>Picture Lions</label>
   <listprice>$10.35</listprice>
   <manufacturer>Picture Lions</manufacturer>
   <publisher>Picture Lions</publisher>
   <readinglevel>Ages 9-12</readinglevel>
   <releasedate/>
   <publicationdate>1983-03-24</publicationdate>
   <studio>Picture Lions</studio>
   <edition/>
   <dewey/>
   <numberofpages>64</numberofpages>
  -<dimensions>
     <height>39</height>
     <width>638</width>
     <length>866</length>
     <weight>35</weight>
   </dimensions>
  -<reviews>
    -<review>
       <date>1996-09-17</date>
       <summary>A FOREVER CHERISHED TREASURE</summary>
      -<content>
         My daughter received this book as a gift for her 5th Birthday. One year later it is
         chocked full of information about her. Each page requires the child to write or draw
         things about their house, school, friends, family, clothes, food, toys, etc. I am surprised
         by the LOW cost of this treasure. It is a great activity book for any age child and once it's
         completed, it provides on-going entertainment. I wish I had this book when I was a child!
         It makes a great gift for any occasion!
       </content>
       <rating>5</rating>
       <totalvotes>4</totalvotes>
       <helpfulvotes>4</helpfulvotes>
     </review>
```

Figure 1: Example of book from the Amazon/LibraryThing collection in XML format

SBS task provides a set of queries called topics where users describe what they are looking for (books for a particular genre, books of particular authors, similar books to those that have been already read, etc.). These requests for recommendations are natural expressions of information needs for a large collection of online book records. The topics are crawled from LibraryThing discussion Forums.

The topic set consists of 680 topics in 2014. Each topic has a narrative description of the information need and other fields as illustrated in Figure 2.

[1] http://social-book-search.humanities.uva.nl/
[2] http://www.amazon.com/
[3] http://www.librarything.com/



```
<topic id="1116">
    <title>Which LISP?</title>
    <mediated_query>introduction book to Lisp</mediated_query>
    <group>Purely Programmers</group>
    <narrative>  It'll be time for me to shake things up and learn a new
        language soon. I had started on Erlang a while back and getting
        back to it might be fun. But I'm starting to lean toward Lisp--
        probably Common Lisp rather than Scheme. Anyone care to recommend
         a good first Lisp book? Would I be crazy to hope that there's
        one out there with an emphasis on using Lisp in a web development
         and/or system administration context? Not that I'm unhappy with
        PHP and Perl, but the best way for me to find the time to learn a
         new language is to use it for my work... </narrative>
</topic>
```

Figure 2: Example of topic, composed with multiple fields to describe user's need(s)

4. RETRIEVAL MODELS

This section describes the retrieval models we used for book recommendation and their combination.

4.1 InL2 of Divergence From Randomness

We used InL2, Inverse Document Frequency model with Laplace after-effect and normalization 2. This model has been used with success in different works [3,6,10,26]. InL2 is a DFR-based model (Divergence From Randomness) based on the Geometric distribution and Laplace law of succession.

4.2 Sequential dependence Model of Markov Random Field

Language models are largely used in Document Retrieval search for book recommendation [6, 7]. Metzler and Croft proposed Markov Random Field (MRF) model [18, 20] that integrates multi-word phrases in the query. Specifically, we used the Sequential Dependence Model (SDM), which is a special case of MRF. In this models co-occurrence of query terms is taken into consideration. SDM builds upon this idea by considering combinations of query terms with proximity constraints which are: single term features (standard unigram language model features, f_T), exact phrase features (words appearing in sequence, f_O) and unordered window features (require words to be close together, but not necessarily in an exact sequence order, f_U).

Finally, documents are ranked according to the following scoring function:

$$SDM(Q, D) = \lambda_T \sum_{q \in Q} f_T(q, D) +$$

$$+\lambda_O \sum_{i=1}^{|Q|-1} f_O(q_i, q_i + 1, D)$$

$$+\lambda_U \sum_{i=1}^{|Q|-1} f_U(q_i, q_i + 1, D)$$

Where feature weights are set based on the authorâĂŹs recommendation ($\lambda_T = 0.85$, $\lambda_O = 0.1$, $\lambda_U = 0.05$) in [7]. f_T , f_O and f_U are the log maximum likelihood estimates of query terms in document D, computed over the target collection using a Dirichlet smoothing. We applied this model to the queries using Indri[4] Query Language[5].

4.3 Combining Search Systems

Combining the output of many search systems, in contrast to using just a single one improves the retrieval effectiveness as proved in [5] where Belkin combined the results of probabilistic with vector space models. On the basis of this approach, In our work, we combined the probabilistic model, InL2 with language model SDM. This combination takes into account both the informativeness of query terms and their dependencies in the document collection. Each retrieval model uses different weighting schemes therefore the scores should be normalized. We used the maximum and minimum scores according to Lee's formula [15].

$$normalizedScore = \frac{oldScore - minScore}{maxScore - minScore}$$

It has been shown in [6] that InL2 and SDM models have different levels of retrieval effectiveness, thus it is necessary to weight individual model scores depending on their overall performance. We used an interpolation parameter (α) that we varied to improve retrieval effectiveness.

5. GRAPH MODELING

In [17], the authors have exploited networks defined by automatically-generated content-similarity links for document retrieval. We provided document analysis to find new way to link them. In our case, we exploited a special type of similarity based on several factors. This similarity is provided by Amazon and corresponds to "Similar Products" given generally for each book. The degree of similarity depends on social information like: number of clicks or purchases and content-based information like book attributes (book description, book title, etc.). The exact formula used by Amazon to combine social and content based information to compute similarity is proprietary. The idea behind this linking method is that documents linked with such type of similarity, the probability that they are in the same context is higher than if they are not connected.

To perform data modeling into DGD, we extracted the "Similar Products" links between documents in order to construct the graph structure. Once used it to enrich results from the retrieval models, in the same spirit as pseudo-relevance-feedback. Each node in the DGD represents document (Amazon description of book), and has set of properties:

- *ID*: book's ISBN
- *content* : book description that include many other properties (title, product description, author(s), users' tags, content of reviews, etc.)
- *MeanRating* : average of ratings attributed to the book

[4] http://www.lemurproject.org/indri/

[5] http://www.lemurproject.org/lemur/IndriQueryLanguage.php

- PR : book's PageRank

Edges in the DGD are directed and correspond to Amazon similarity, so given nodes $\{A, B\} \in S$, if A points to B, B is suggested as Similar Product to A. In the Figure 3, we show an example of DGD, network of documents. The DGD network contains 1 645 355 nodes (89.86% of nodes are within the collection and the rest are outside) and 6 582 258 edges.

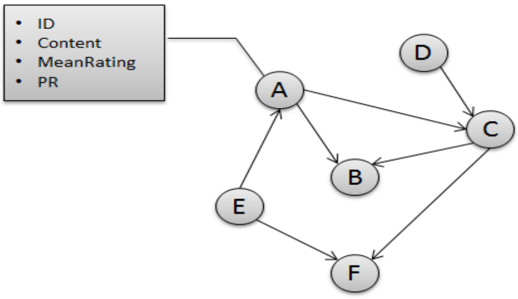


Figure 3: Example of Directed Graph of Documents

Figure 4 shows the general architecture of our document retrieval system with two-level document search. In this system, the *IR Engine* finds all relevant documents for user's query. Then, the *Graph Search* module selects resulting document returned by *Graph Analysis* module. The *Graph Structured Data* is a network constructed using *Social Information Matrix* and enriched by *Compute PageRank* module. The *Social Information Matrix* is constructed by two modules: "Ratings" and "Similar Products" Extraction from the *Data Collection* that contains description books in XML format. *Scoring Ranking* module combines scores of documents resulting from *IR Engine* and *Graph Analysis* modules and reranks them.

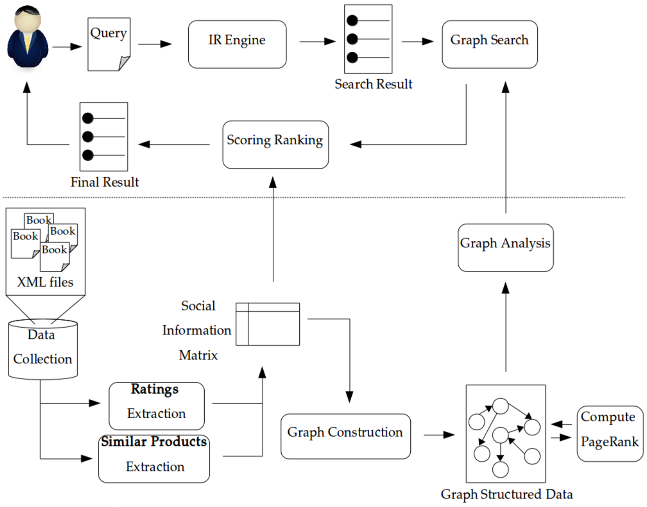


Figure 4: Architecture of document retrieval approach based on graph of documents

In this section, the collection of documents is denoted by C. In C, each document d has a unique ID. The set of queries called topics is denoted by T, the set $D_{init} \subset C$ refers to the documents returned by the initial retrieval model. $StartingNode$ identifies a document from D_{init} used as input to the graph processing algorithms in the DGD. The set of documents present in the graph is denoted by S. D_{t_i} indicates the documents retrieved for topic $t_i \in T$.

5.1 Our Approach

The DGD network contains useful information about documents that can be exploited for document retrieval. Our approach is based, first on results of a traditional retrieval engine, then on the DGD network to find new documents. The idea is to suppose that the suggestions given by Amazon can be relevant to the user queries.

Algorithm 1 takes as inputs: D_{init} returned list of documents for each topic by the retrieval techniques described in Section 3, DGD network and parameter β which is the number of the top selected $StartingNode$ from D_{init} denoted by $D_{StartingNodes}$. We fixed β to 100 (10% of the returned list for each topic). The algorithm returns a list of recommendations for each topic denoted by "D_{final}". It processes topic by topic, and extracts the list of all neighbors for each $StartingNode$. It performs mutual Shortest Paths computation between all selected $StartingNode$ in DGD. The two lists (neighbors and nodes in computed Shortest Paths) are concatenated after that all duplicated nodes are deleted. The set of documents in returned list is denoted by D_{graph}. A second concatenation is performed between initial list of documents and D_{graph} (all duplications are deleted) in new final list of retrieved documents, D_{final} reranked using different reranking schemes.

Algorithm 1 Retrieving based on DGD feedback

```
$D_{init} \leftarrow$ Retrieving Documents for each $t_i \in T$
**for** each $D_{t_i} \in D_{init}$ **do**
    $D_{StartingNodes} \leftarrow$ first $\beta$ documents $\in D_{t_i}$
    **for** each $StartingNode$ **in** $D_{StartingNodes}$ **do**
        $D_{graph} \leftarrow D_{graph} + neighbors(StartingNode, DGD)$
        $D_{SPnodes} \leftarrow$ **all** $D \in ShortestPath(StartingNode, D_{StartingNodes}, DGD)$
        $D_{graph} \leftarrow D_{graph} + D_{SPnodes}$
        Delete all duplications from $D_{graph}$
    $D_{final} \leftarrow D_{final} + (D_{t_i} + D_{graph})$
Delete all duplications from $D_{final}$
Rerank $D_{final}$
```

Figure 5 shows an illustration of the document retrieval approach based on DGD feedback.

6. EXPERIMENTS AND RESULTS

In this section, we describe the experimental setup we used for our experiments. Furthermore, we present the different reranking schemes used in previously defined approaches. We discuss the results we achieved by using the InL2 retrieval model, its combination to the SDM model, and retrieval system proposed in our approach that uses the DGD network.

6.1 Experiments setup

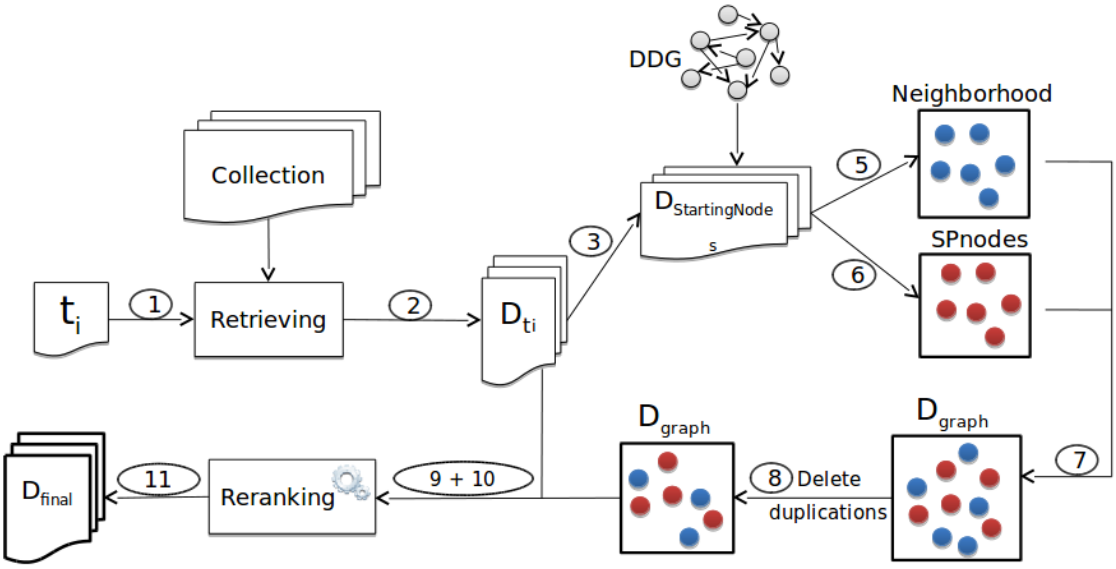


Figure 5: Book retrieval approach based on DGD feedback. Numbers on the arrows refer to the instructions in the Algorithm 1

For our experiments, we used different tools that implement retrieval models and handle the graph processing. First, we used *Terrier* (TERabyte RetrIEveR)[6] *Information Retrieval* framework developed at the University of Glasgow [21–23]. Terrier is a modular platform for rapid development of large-scale IR applications. It provides indexing and retrieval functionalities. It is based on DFR framework and we used it to deploy InL2 model described in section 4.1. Further information about Terrier can be found at http://ir.dcs.gla.ac.uk/terrier.

A preprocessing step was performed to convert INEX SBS corpus into the Trec Collection Format[7], by considering that the content of all tags in each XML file is important for indexing; therefore the whole XML file was transformed on one document identified by its ISBN. Thus, we just need two tags instead of all tags in XML, the ISBN and the whole content (named text).

Secondly, *Indri*[8], *Lemur Toolkit for Language Modeling and Information Retrieval* was used to carry out a language model (SDM) described in section 4.2. Indri is a framework that provides state-of-the-art text search methods and a rich structured query language for big collections (up to 50 million documents). It is a part of the Lemur project and developed by researchers from UMass and Carnegie Mellon University. We used Porter stemmer and performed Bayesian smoothing with Dirichlet priors (Dirichlet prior $\mu = 1500$).

In section 5.1, we have described our approach based on DGD which includes graph processing. We used NetworkX[9] tool of Python to perform shortest path computing, neighborhood extraction and PageRank calculation.

To evaluate the results of retrieval systems, several measurements have been used for SBS task: Discounted Cumulative Gain (nDCG), the most popular measure in IR [11], Mean Average Precision (MAP) which calculates the mean of average precisions over a set of queries, and other measures: Recip Rank and Precision at the rank 10 (P@10).

6.2 Reranking Schemes

Two approaches were proposed. The first one (see section 4.3) merges the results of two different information retrieval models which are the Language Model (SDM) and DFR model (InL2). For topic t_i, the models give 1000 documents and each retrieved document has an associated score. The linear combination method uses the following formula to calculate final score for each retrieved document d by SDM and InL2 models:

$$S_{final}(d, t_i) = \alpha * S_{InL2}(d, t_i) + (1 - \alpha) * S_{SDM}(d, t_i)$$

Where $S_{InL2}(d, t_i)$ and $S_{SDM}(d, t_i)$ are normalized scores. α is the interpolation parameter set up at 0.8 after several tests on the 2014 topics.

The second approach (described in 5.1) uses the DGD constructed from the “Similar Products” information. The document set returned by the retrieval model are fused to the documents in neighbors set and Shortest Path results. We tested many reranking methods that combine the retrieval model scores and other scores based on social information. For each document in the resulting list, we calculated the following scores:

- **PageRank**, computed using NetworkX tool. It is a well-known algorithm that exploits link structure

[6] http://terrier.org/
[7] http://lab.hypotheses.org/1129
[8] http://www.lemurproject.org/indri/
[9] https://networkx.github.io/

to score the importance of nodes in a graph. Usually, it was been used for hyperlink graphs such as the Web [24]. The values of PageRank are given by the following formula.

$$PR(A) = (1 - d) + d(PR(T_1)/C(T_1) \\ + ... + PR(T_n)/C(T_n))$$

Where document A has documents $T_1 ... T_n$ which point to it (i.e., Similar products). The parameter d is a damping factor set between 0 and 1 (0.85 in our case). $C(A)$ is defined as the number of links going out of page A.

- **Likeliness**, computed from information generated by users (reviews and ratings). It is based on the idea that more the book has a lot of reviews and good ratings, the more interesting it is (it may not be a good or popular book but a book that has a high impact).

$$Likeliness(D) = \log(\#reviews(D)) \times \frac{\sum_{r \in R_D} r}{\#reviews(D)}$$

Where $\#reviews(D)$ is the number of reviews attributed to D, R_D is the set of reviews of D.

The computed scores were normalized using this formula: $normalized_{score} = old_{score}/max_{score}$. After that, to combine the results of retrieval systems and each of normalized scores, an intuitive solution is to weight the retrieval model scores with the previously described scores (normalized PageRank and Likeliness). However, this would favor documents with high PageRank and Likeliness scores even though their content is much less related to the topics.

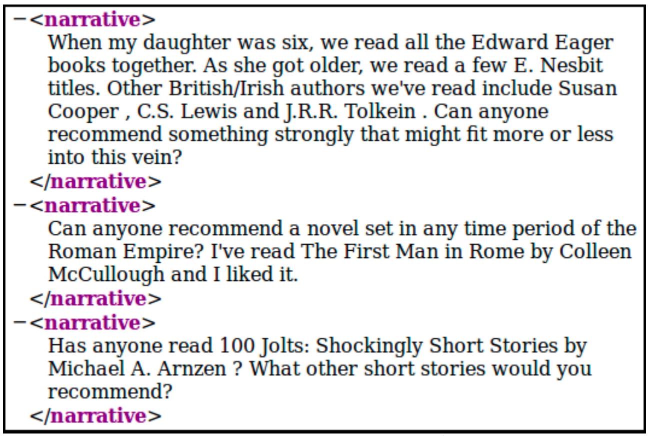
6.3 Results

We used two topic sets provided by INEX SBS task in 2014 (680 topics). The systems retrieve 1000 documents per topic. We assessed the narrative field of each topic and provided automatic classification of the topic set into 2 genres. *Analogue* topics (261) in which users give the already read books (generally, titles and authors) to have similar books. In the second genre "*Non-Analogue*" (356 topics), users describe their needs by defining the thematic, interested field, event, etc. without citing other books. Notify that, 63 topics are ignored because of their ambiguity.

In order to evaluate our IR methodologies described in sections 4.3, 5 we performed retrieving for each topic genre individually. The experimental results, which describe the performance of the different retrieval systems on Amazon/LibraryThing document collection, are shown in Table 1.

As illustrated in Table 1, the system that combines probabilistic model InL2 and the Language Model SDM (InL2_SDM) achieves a significant improvement for each topic set comparing to InL2 model (Baseline) but the improvement is highest for *Non-Analogue* topic set where the content of queries are more explicit than the other topic set. This improvement is mainly due to the increase of the number of relevant documents that are retrieved by both systems.

The results of run InL2_DDG_PR using the *Analogue* topic set confirm that exploiting structured documents and performing reranking with PageRank improves significantly performances but in contrast, it lowers the baseline performances when using the *Non-Analogue* topic set. This can be explained by the fact that *Analogue* topics contain examples of books (Figure 6) which require the use of graph to extract the similar connected books.



```
−<narrative>
    When my daughter was six, we read all the Edward Eager
    books together. As she got older, we read a few E. Nesbit
    titles. Other British/Irish authors we've read include Susan
    Cooper , C.S. Lewis and J.R.R. Tolkein . Can anyone
    recommend something strongly that might fit more or less
    into this vein?
  </narrative>
−<narrative>
    Can anyone recommend a novel set in any time period of the
    Roman Empire? I've read The First Man in Rome by Colleen
    McCullough and I liked it.
  </narrative>
−<narrative>
    Has anyone read 100 Jolts: Shockingly Short Stories by
    Michael A. Arnzen ? What other short stories would you
    recommend?
  </narrative>
```

Figure 6: Examples of narratives in *Analogue* topics

Using Likeliness scores (in InL2_DGD_MnRtg) to rerank retrieved documents decreases significantly the baseline efficiency for the two topic sets. This means that ratings given by users don't provide any improvement for the reranking performances.

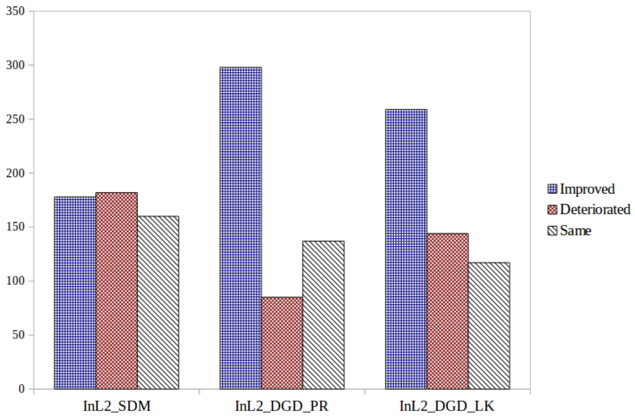


Figure 7: Histograms that demonstrate and compare the number of improved, deteriorated and same results' topics using the proposed approaches for MAP measure. (Baseline: InL2)

Figure 7 compares the number of improved, deteriorated and same results' topics between the baseline (InL2) and the proposed retrieval systems in term of MAP measure. The proposed systems based on DGD graph provide the highest number of improved topics compared with the combination of IR systems. More precisely, using PageRank to rerank document produces better results in term of improved topics. This results prove the positive impact of linked structure on document retrieval systems for book recommendation.

The depicted results confirm that we are starting with competitive baseline, suggesting that improvements contribute

Table 1: Experimental results. The runs are ranked according to nDCG@10. (∗) denotes significance according to Wilcoxon test [9]. In all cases, all of our tests produced two-sided p-value, $\alpha = 0.05$.

	Analogue **topics**				*Non-Analogue* **topics**			
Run	nDCG@10	Recip Rank	MAP	P@10	nDCG@10	Recip Rank	MAP	P@10
InL2	**0.1099**	**0.267**	**0.072**	**0.078**	**0.138**	**0.207**	**0.117**	**0.0579**
InL2_SDM	0.1115 (+1%*)	0.271 (+1%*)	0.073 (+0.6%)	0.079 (+1%*)	0.147(+6%*)	0.222(+7%*)	0.124(+5%*)	0.0630(+8%*)
InL2_DGD_PR	0.1111 (+1%*)	0.277 (+3%*)	0.068 (−5%*)	0.082 (+12%)	0.127(−7%*)	0.206(−0.6%*)	0.102(−12%*)	0.0570(−1%*)
InL2_DGD_LK	0.1043 (−5%)	0.275 (+2%)	0.064(−11%*)	0.082(+5%)	0.130(−5%)	0.214(+3%*)	0.100(−14%*)	0.0676(+16%)

by combining output retrieval systems and social link analysis are indeed meaningful.

7. HUMANITIES AND SOCIAL SCIENCES COLLECTION: GRAPH MODELING AND RECOMMENDATION

We tested the proposed approach of recommendation based on linked documents on Revues.org[10] collection. Revues.org is one of the four platforms of OpenEdition[11] portal dedicated to electronic resources in the humanities and social sciences (books, journals, research blogs, and academic announcements). Revues.org was founded in 1999 and today it hosts over 400 online journals, i.e. 149000 articles, proceedings ans editorials.

We built a network of documents from ASp[12] journal. It publishes research articles, publication listings and reviews related to the field of English for Specific Purposes (ESP) for both teaching and research. The network contains 500 documents and 833 relationships which represent bibliographic citations. Each relationship is constructed using BILBO [12], the reference parsing software. BILBO is constructed with annotated corpora from Digital Humanities articles from OpenEdition Revues.org platform. It automatic annotates bibliographic references in the bibliography section of each document and obtains the corresponding DOI (Digital Object Identifier) via CrossRef[13] API if such an identifier exists.

Each node in the citation network have a set of properties (*ID* which is its URL, *type*, it can be article, editorial, review of book, etc., and readers' clicks number that we called *popularity*). The recommender system applied on this network takes as input user query, generally a small set of short keywords, and performs retrieval step using Solr[14] search engine. The system extend the returned results with documents in the citation network by using graph algorithms (neighborhood search and shortest path algorithm) as described in section5.1. After that, we rerank documents according to the popularity property of each document.

We tested the system manually for a small set of user queries, and found that for most queries, the results were satisfying.

8. CONCLUSION AND FUTURE WORK

In this paper, we proposed and evaluated approaches of document retrieval in the context of book recommendation. We used the test collection of CLEF Labs Social Book Search track and the proposed topics in 2014 divided into two classes *Analogue* and "*Non-Analogue*".

We presented the first approach that combines the outputs of probabilistic model (InL2) and Language Model (SDM) using a linear interpolation after normalizing scores of each retrieval system. We have shown a significant improvement of baseline results using this combination.

A novel approach was proposed, based on Directed Graph of Documents (DGD) constructed from social relationships. It exploits link structure to enrich the returned document list by traditional retrieval model (InL2). We performed a reranking method using PageRank and Likeliness of each retrieved document.

In the future, we would like to construct an evaluation corpora from Revues.org collection and develop an evaluation process similar to that of INEX SBS task. Another interesting extension of our work would be using the learning to rank techniques to automatically adjust the settings of re-ranking parameters.

9. ACKNOWLEDGMENT

This work was supported by the French program Investissements d'Avenir FSN and the French Région PACA under the projects InterTextes and Agoraweb.

10. REFERENCES

[1] M. Abolhassani and N. Fuhr. Applying the divergence from randomness approach for content-only search in XML documents. pages 409–419, 2004.

[2] G. Amati and C. J. Van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trans. Inf. Syst.*, 20(4):357–389, Oct. 2002.

[3] G. Amati and C. J. van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trans. Inf. Syst.*, 20(4):357–389, October 2002.

[4] T. Beckers, N. Fuhr, N. Pharo, R. Nordlie, and K. N. Fachry. Overview and results of the INEX 2009 interactive track. In *Research and Advanced Technology for Digital Libraries, 14th European Conference, ECDL 2010, Glasgow, UK, September 6-10, 2010. Proceedings*, pages 409–412, 2010.

[5] N. J. Belkin, P. B. Kantor, E. A. Fox, and J. A. Shaw. Combining the evidence of multiple query representations for information retrieval. *Inf. Process. Manage.*, 31(3):431–448, 1995.

[6] C. Benkoussas, H. Hamdan, S. Albitar, A. Ollagnier, and P. Bellot. Collaborative filtering for book recommandation. In *Working Notes for CLEF 2014*

[10] http://www.revues.org/

[11] http://www.openedition.org

[12] http://www.openedition.org/6457

[13] http://www.crossref.org/

[14] http://lucene.apache.org/solr/

Conference, Sheffield, UK, September 15-18, 2014., pages 501–507, 2014.

[7] L. Bonnefoy, R. Deveaud, and P. Bellot. Do social information help book search? In P. Forner, J. Karlgren, and C. Womser-Hacker, editors, *CLEF (Online Working Notes/Labs/Workshop)*, 2012.

[8] H. Bouchard and J.-Y. Nie. ModÃĺles de langue appliquÃl's Ãă la recherche d'information contextuelle. In *CORIA*, pages 213–224. UniversitÃl' de Lyon, 2006.

[9] W. B. Croft. *Organizing and searching large files of document descriptions.* PhD thesis, Cambridge University, 1978.

[10] R. GuillÃl'n. Gir with language modeling and dfr using terrier. In C. Peters, T. Deselaers, N. Ferro, J. Gonzalo, G. Jones, M. Kurimo, T. Mandl, A. PeÃśas, and V. Petras, editors, *Evaluating Systems for Multilingual and Multimodal Information Access*, volume 5706 of *Lecture Notes in Computer Science*, pages 822–829. Springer Berlin Heidelberg, 2009.

[11] K. Järvelin and J. Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In E. Yannakoudakis, N. Belkin, P. Ingwersen, and M.-K. Leong, editors, *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2000)*, pages 41–48, New York, NY, USA, 2000. ACM.

[12] Y.-M. Kim, P. Bellot, E. Faath, and M. Dacos. Automatic annotation of bibliographical references in digital humanities books, articles and blogs. In G. Kazai, C. Eickhoff, and P. Brusilovsky, editors, *BooksOnline*, pages 41–48. ACM, 2011.

[13] M. Koolen, T. Bogers, J. Kamps, G. Kazai, and M. Preminger. Overview of the INEX 2014 social book search track. In *Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15-18, 2014.*, pages 462–479, 2014.

[14] O. Kurland and L. Lee. PageRank without hyperlinks: Structural re-ranking using links induced by language models. In *Proceedings of SIGIR*, pages 306–313, 2005.

[15] J. H. Lee. Combining multiple evidence from different properties of weighting schemes. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '95, pages 180–188, New York, NY, USA, 1995. ACM.

[16] J. Lin. Pagerank without hyperlinks: Reranking with pubmed related article networks for biomedical text retrieval. *BMC Bioinformatics*, 9(1), 2008.

[17] J. Lin. Pagerank without hyperlinks: Reranking with pubmed related article networks for biomedical text retrieval. *BMC Bioinformatics*, 9(1), 2008.

[18] D. Metzler and W. B. Croft. Combining the language model and inference network approaches to retrieval. *Inf. Process. Manage.*, 40(5):735–750, 2004.

[19] D. Metzler and W. B. Croft. A markov random field model for term dependencies. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 472–479, New York, NY, USA, 2005. ACM.

[20] D. Metzler and W. B. Croft. A markov random field model for term dependencies. In R. A. Baeza-Yates, N. Ziviani, G. Marchionini, A. Moffat, and J. Tait, editors, *SIGIR*, pages 472–479. ACM, 2005.

[21] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and C. Lioma. Terrier: A High Performance and Scalable Information Retrieval Platform. In *Proceedings of ACM SIGIR'06 Workshop on Open Source Information Retrieval (OSIR 2006)*, 2006.

[22] I. Ounis, G. Amati, P. V., B. He, C. Macdonald, and Johnson. Terrier Information Retrieval Platform. In *Proceedings of the 27th European Conference on IR Research (ECIR 2005)*, volume 3408 of *Lecture Notes in Computer Science*, pages 517–519. Springer, 2005.

[23] I. Ounis, C. Lioma, C. Macdonald, and V. Plachouras. Research directions in terrier: a search engine for advanced retrieval on the web. *Novatica/UPGRADE Special Issue on Web Information Access*, 2007.

[24] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. In *Proceedings of the 7th International World Wide Web Conference*, pages 161–172, Brisbane, Australia, 1998.

[25] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[26] V. Plachouras, B. He, and I. Ounis. University of glasgow at trec 2004: Experiments in web, robust, and terabyte tracks with terrier. In E. M. Voorhees and L. P. Buckland, editors, *TREC*, volume Special Publication 500-261. National Institute of Standards and Technology (NIST), 2004.

[27] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proc. SIGIR*, 1998.

[28] S. E. Robertson, C. J. van Rijsbergen, and M. F. Porter. Probabilistic models of indexing and searching. In *SIGIR*, pages 35–56, 1980.

[29] F. Song and W. Croft. A general language model for information retrieval. In *Proceedings of the SIGIR Conference on Information Retrieval*, 1999.

[30] T. Tao, X. Wang, Q. Mei, and C. Zhai. Language model information retrieval with document expansion. In R. C. Moore, J. A. Bilmes, J. Chu-Carroll, and M. Sanderson, editors, *HLT-NAACL*. The Association for Computational Linguistics, 2006.

[31] C. Zhai. *Statistical Language Models for Information Retrieval.* Synthesis Lectures on Human Language Technologies. Morgan and Claypool Publishers, 2008.

The Continuous Cold Start Problem in e-Commerce Recommender Systems

Lucas Bernardi[1], Jaap Kamps[2], Julia Kiseleva[3], Melanie J.I. Mueller[1]

[1]Booking.com, Amsterdam, Netherlands. Email: {lucas.bernardi, melanie.mueller}@booking.com
[2]University of Amsterdam, Amsterdam, Netherlands. Email: kamps@uva.nl
[3]Eindhoven University of Technology, Eindhoven, Netherlands. Email: j.kiseleva@tue.nl

ABSTRACT

Many e-commerce websites use recommender systems to recommend items to users. When a user or item is new, the system may fail because not enough information is available on this user or item. Various solutions to this 'cold-start problem' have been proposed in the literature. However, many real-life e-commerce applications suffer from an aggravated, recurring version of cold-start even for known users or items, since many users visit the website rarely, change their interests over time, or exhibit different personas. This paper exposes the *Continuous Cold Start* (CoCoS) problem and its consequences for content- and context-based recommendation from the viewpoint of typical e-commerce applications, illustrated with examples from a major travel recommendation website, Booking.com.

General Terms

CoCoS: continuous cold start

Keywords

Recommender systems, continous cold-start problem, industrial applications

1. INTRODUCTION

Many e-commerce websites are built around serving personalized recommendations to users. Amazon.com recommends books, Booking.com recommends accommodations, Netflix recommends movies, Reddit recommends news stories, etc. Two examples of recommendations of accomodations and destinations at Booking.com are shown in Figure 1. This widescale adoption of recommender systems online, and the challenges faced by industrial applications, have been a driving force in the development of recommender systems. The research area has been expanding since the first papers on collaborative filtering in the 1990s [12, 16]. Many different recommendation approaches have been developed since then, in particular content-based and hybrid approaches have supplemented the original collaborative filtering techniques [1].

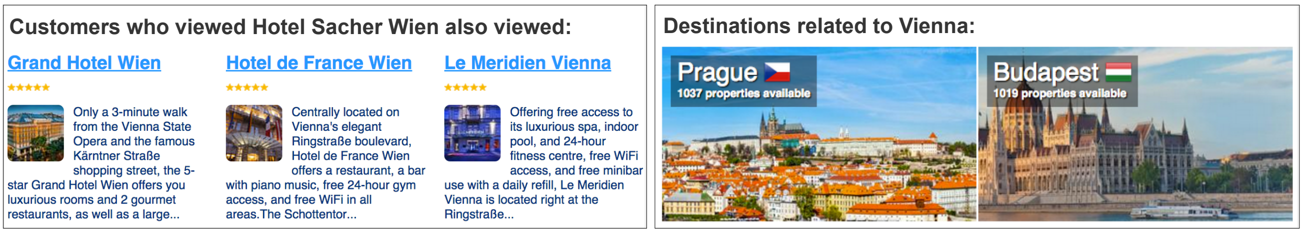
In the most basic formulation, the task of a recommender system is to predict ratings for items that have not been seen by the user. Using these predicted ratings, the system decides which new items to recommend to the user. Recommender systems base the prediction of unknown ratings on past or current information about the users and items, such as past user ratings, user profiles, item descriptions etc. If this information is not available for new users or items, the recommender system runs into the so-called *cold-start problem*: It does not know what to recommend until the new, 'cold', user or item has 'warmed-up', i.e. until enough information has been generated to produce recommendations. For example, which accomodations should be recommended to someone who visits Booking.com for the first time? If the recommender system is based on which accomodations users have clicked on in the past, the first recommendations can only be made after the user has clicked on a couple of accomodations on the website.

Several approaches have been proposed and successfully applied to deal with the cold-start problem, such as utilizing baselines for cold users [8], combining collaborative filtering with content-based recommenders in hybrid systems [14], eliciting ratings from new users [11], or, more recently, exploiting the social network of users [6, 15]. In particular, content-based approaches have been very successful in dealing with cold-start problems in collaborative filtering [3, 4, 13, 14].

These approaches deal explicitly with cold users or items, and provide a 'fix' until enough information has been gathered to apply the core recommender system. Thus, rather than providing unified recommendations for cold and warm users, they temporarily bridge the period during which the user or item is 'cold' until it is 'warm'. This can be very successful in situations in which there are no warm users [3], or in situations when the warm-up period is short and warmed-up users or items stay warm.

However, in many practical e-commerce applications, users or items remain cold for a long time, and can even 'cool down' again, leading to a *continuous* cold-start (CoCoS). In the example of Booking.com, many users visit and book infrequently since they go on holiday only once or twice a year, leading to a prolonged cold-start and extreme sparsity of collaborative filtering matrices, see Fig. 2 (top). In addition, even warm long-term users can cool down as they change their needs over time, e.g. going from booking youth hostels for road trips to resorts for family vacations. Such cool-

CBRecSys 2015, September 20, 2015, Vienna, Austria.



Customers who viewed Hotel Sacher Wien also viewed:

Grand Hotel Wien

★★★★★

Only a 3-minute walk from the Vienna State Opera and the famous Kärntner Straße shopping street, the 5-star Grand Hotel Wien offers you luxurious rooms and 2 gourmet restaurants, as well as a large...

Hotel de France Wien

★★★★★

Centrally located on Vienna's elegant Ringstraße boulevard, Hotel de France Wien offers a restaurant, a bar with piano music, free 24-hour gym access, and free WiFi in all areas.The Schottentor...

Le Meridien Vienna

★★★★★

Offering free access to its luxurious spa, indoor pool, and 24-hour fitness centre, free WiFi access, and free minibar use with a daily refill, Le Meridien Vienna is located right at the Ringstraße...

Destinations related to Vienna:

Prague

1037 properties available

Budapest

1019 properties available

Figure 1: Examples of recommender systems on Booking.com. User-to-user collaborative filtering (left): recommend accomodations viewed by similar users to a user who just looked at 'Hotel Sacher Wien'. Item-to-item content-based recommendations (right): recommend destinations similar to a particular destination, Vienna.

downs can happen more frequently and rapidly for users who book accommodations for different travel purposes, e.g. for leisure holidays and business trips as shown in Fig. 2 (bottom). These continuous cold-start problems are rarely addressed in the literature despite their relevance in industrial applications. Classical approaches to the cold-start problem fail in the case of CoCoS, since they assume that users warm up in a reasonable time and stay warm after that.

In the remainder of the paper, we will elaborate on how CoCoS appears in e-commerce websites (Sec. 2), outline some approaches to the CoCoS problem (Sec. 3), and end with a discussion about possible future directions (Sec. 4).

2. CONTINUOUS COLD-START

Cold-start problems can in principle arise on both the user side and the items side.

2.1 User Continuous Cold-Start

We first focus on the user side of CoCoS, which can arise in the following cases:

Classical cold-start / sparsity: new or rare users

Volatility: user interest changes over time

Personas: user has different interests at different, possibly close-by points in time

Identity: failure to match data from the same user

All cases arise commonly in e-commerce websites. New users arrive frequently (classical cold-start), or may appear new when they don't log in or use a different device (failed identity match). Some websites are prone to very low levels of user activity when items are purchased only rarely, such as travel, cars etc., leading to sparsity problems for recommender systems. Most users change their interests over time (volatility), e.g. movie preferences evolve, or travel needs change. On even shorter timescales, users have different personas. Depending on their mood or their social context, they might be interested in watching different movies. Depending on the weather or their travel purpose, they may want to book different types of trips, see Figure 2 for examples from Booking.com.

These issues arise for collaborative filtering as well as content-based or hybrid approaches, since both user ratings or activities as well user profiles might be missing, become outdated over time, or not be relevant to the current user persona.

2.2 Item Continuous Cold-Start

In a symmetric way, these CoCoS problems also arise for items:

Classical cold-start / sparsity: new or rare items

Volatility: item properties or value changes over time

Personas: item appeals to different types of users

Identity: failure to match data from the same item

New items appear frequently in e-commerce catalogues, as shown in Figure 3 for accommodations at Booking.com. Some items are interesting only to niche audiences, or sold only rarely, for example books or movies on specialized topics. Items can be volatile if their properties change over time, such as s phone that becomes outdated once a newer model is released, or a hotel that undergoes a renovation. In the context of news or conversions, item volatility is also known as topic drift [9]. Figure 3 on the right shows fluctuations of the review score of a hotel at Booking.com. Some items have different 'personas' in that they target several user groups, such as a hotel that caters to business as well as leisure travellers. When several sellers can add items to an e-commerce catalogue, or when several catalogues are combined, correctly matching items can be problematic (identity problem).

3. ADDRESSING COLD-START

Many approaches have been proposed to deal with the classical cold-start problem of new or rare users or items [11]. However, they mostly fail to address the more difficult CoCoS.

The most popular strategy to address the classic cold-start problem is the hybrid approach where collaborative filtering and content-based models are combined, see [14] as an example. If one of the two method fails due to a new user or item, the other method is used to 'fill-in'. The most basic assumption is that similar users will like similar items. Similarity of users is measured by their purchase history when warm, and by their user profile when cold. Conversely, similarities between items is computed by the set of users that purchased them when warm, and by their content when cold. In CoCoS, users change their interests, so both collaborative filtering and user-profile-based approaches can fail, since looking at the past and similarities can be misleading. Items also suffer from volatility, although to a lesser degree, which makes the standard hybrid approach also problematic for

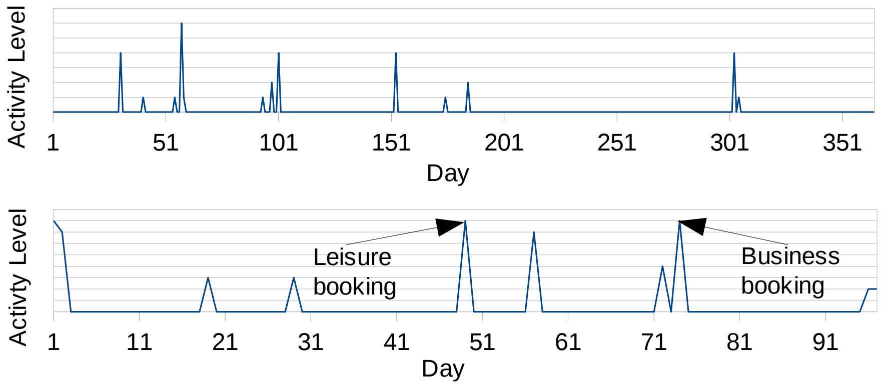


Figure 2: Continuously cold users at Booking.com. Activity levels of two randomly chosen users of Booking.com over time. The top user exhibits only rare activity throughout a year, and the bottom user has two different personas, making a leisure and a business booking, without much activity inbetween.

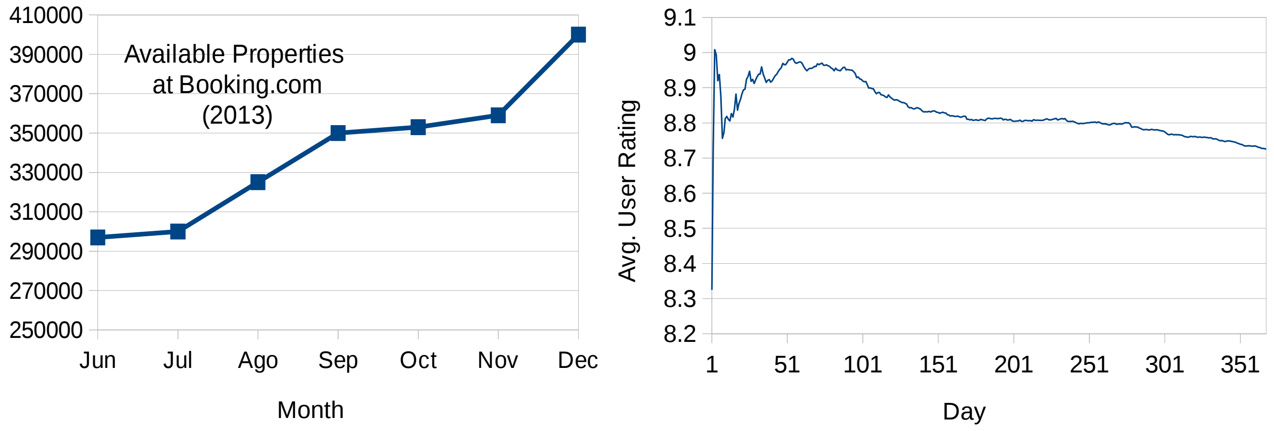


Figure 3: Continuously cold items at Booking.com. Thousands of new accommodations are added to Booking.com every month (left). The user ratings of a hotel can change continuously (right).

items. Hybrid approaches also ignore the issue of multiple personas.

Although, to our knowledge, the continuous cold-start problem as defined in this work has not been directly addressed in the literature, several approaches are promising.

Tang et al. [19] propose a context-aware recommender system, implemented as a contextual multi-armed bandits problem. Although the authors report extensive offline evaluation (log based and simulation based) with acceptable CTR, no comparison is made from a cold-start problem standpoint.

Sun et al. [18] explicitly attack the user volatility problem. They propose a dynamic extension of matrix factorization where the user latent space is modeled by a state space model fitted by a Kalman filter. Generative data presenting user preference transitions is used for evaluation. Improvements of RMSE when compared to timeSVD [10] are reported. Consistent results are reported in [5], after offline evaluation using real data.

Tavakol and Brefeld [20] propose a topic driven recommender system. At the user session level, the user intent is modeled as a topic distribution over all the possible item attributes. As the user interacts with the system, the user intent is predicted and recommendations are computed using the corresponding topic distribution. The topic prediction is solved by factored Markov decision processes. Evaluation on an e-commerce data set shows improvements when compared to collaborative filtering methods in terms of average rank.

4. DISCUSSION

In this manuscript, we have described how CoCoS, the continuous cold-start problem, is a common issue for e-commerce applications. Industrial recommender systems do not only have to deal with 'cold' (new or rare) users and items, but also with known users or items that repeatedly 'cool down'. Reasons for the recurring cool-downs include the volatility in user interests or item values, different personas depending on user context or item target audience, or identification problems due to logged-out users or items from different catalogues. Despite the practical relevance of CoCoS, common literature approaches do not deal well with this issue.

We consider several directions as particularly promising to deal with CoCoS. Traditional approaches to solve cold-start problems try to employ collaborative filtering based on pseudo or inferred clicks. Recommendations based on social networks are an interesting new development that can supplement missing information based on the social graph. For example, recommendations based on Facebook likes are proposed in [15]. Beyond the difficulty to get access to social data, the application to user volatility or multiple personas remains challenging. Online user intent prediction can be used to estimate a user's current profile on the fly. When a user visits the website, his browsing behavior is used to estimate his intent after a few clicks, which are then used to compute recommendations accordingly. However, this still delays recommendations until enough clicks have occurred, which can be problematic if quick recommendations are needed. For example, in last-minute bookings, users may be pressed to book an accommodation quickly, leading to very short sessions.

More promising approaches employ content based or contextual recommendation. Content based recommendations can be very effective based on very little signal: just an initial query or single interaction can be exploited to find an initial item or set of items and exploit relations between items to make effective recommendations. In particular context aware recommendations are one of the most promising strategies when it comes to solving CoCoS. In this setup, recommendations are computed based on the current context of the current visitor and the behaviour of other users in similar contexts [see 2, 7, 17] for examples. Context is defined as a set of features such as location, time, weather, device, etc. Often this data is readily available in most commercial implementations of recommender systems. This approach naturally addresses sparsity by clustering users into contexts. Since context is determined in a per-action basis, user volatility and multiple personas can be addressed robustly. On the other hand, context aware recommenders cannot address the item side of the problem and they might also suffer from cold-start problems in the case of a cold context that has never seen before by the system.

References

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17:734–749, 2005.

[2] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Recommender Systems Handbook*, pages 217–253, 2011.

[3] M. Aharon, N. Aizenberg, E. Bortnikov, R. Lempel, R. Adadi, T. Benyamini, L. Levin, R. Roth, and O. Serfaty. OFF-set: One-pass factorization of feature sets for online recommendation in persistent cold start settings. In *Proceedings of the 7th ACM Conference on Recommender Systems*, pages 375–378, 2013.

[4] S. Bykau, F. Koutrika, and Y. Velegrakis. Coping with the persistent cold-start problem. In *Personalized Access, Profile Management, and Context Awareness in Databases*, 2013.

[5] F. C. T. Chua, R. J. Oentaryo, and E.-P. Lim. Modeling temporal adoptions using dynamic matrix factorization. In *IEEE 13th International Conference on Data Mining (ICDM)*, pages 91–100, 2013.

[6] I. Guy, N. Zwerdling, D. Carmel, I. Ronen, E. Uziel, S. Yogev, and S. Ofek-Koifman. Personalized recommendation of social software items based on social relations. In *Proceedings of the Third ACM Conference on Recommender Systems*, pages 53–60, 2009.

[7] A. Hawalah and M. Fasli. Utilizing contextual ontological user profiles for personalized recommendations. *Expert Syst. Appl. (ESWA)*, 41:4777–4797, 2014.

[8] D. Kluver and J. A. Konstan. Evaluating recommender behavior for new users. In *Proceedings of the 8th ACM Conference on Recommender Systems*, pages 121–128, 2014.

[9] D. Knights, M. Mozer, and N. Nicolov. Detecting topic drift with compound topic models. In *Proceedings of the Third International ICWSM Conference*, pages 242–245, 2009.

[10] Y. Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53:89–97, 2010.

[11] A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl. Getting to know you: Learning new user preferences in recommender systems. In *Proceedings of the 7th International Conference on Intelligent User Interfaces*, pages 127–134, 2002.

[12] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, pages 175–186, 1994.

[13] M. Saveski and A. Mantrach. Item cold-start recommendations: Learning local collective embeddings. In *Proceedings of the 8th ACM Conference on Recommender Systems*, pages 89–96, 2014.

[14] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260, 2002.

[15] S. Sedhain, S. Sanner, D. Braziunas, L. Xie, and J. Christensen. Social collaborative filtering for cold-start recommendations. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 345–348, 2014.

[16] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating 'word of mouth'. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 210–217, 1995.

[17] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, and A. Hanjalic. Cars2: Learning context-aware representations for context-aware recommendations. In *Proceeding of CIKM*, pages 291–300, 2014.

[18] J. Z. Sun, K. R. Varshney, and K. Subbian. Dynamic matrix factorization: A state space approach. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1897–1900, 2012.

[19] L. Tang, Y. Jiang, L. Li, and T. Li. Ensemble contextual bandits for personalized recommendation. In *Proceedings of the 8th ACM Conference on Recommender Systems*, pages 73–80, 2014.

[20] M. Tavakol and U. Brefeld. Factored mdps for detecting topics of user sessions. In *Proceedings of the 8th ACM Conference on Recommender Systems*, pages 33–40, 2014.

Metadata Embeddings for User and Item Cold-start Recommendations

Maciej Kula
Lyst
maciej.kula@lyst.com

ABSTRACT

I present a hybrid matrix factorisation model representing users and items as linear combinations of their content features' latent factors. The model outperforms both collaborative and content-based models in cold-start or sparse interaction data scenarios (using both user and item metadata), and performs at least as well as a pure collaborative matrix factorisation model where interaction data is abundant. Additionally, feature embeddings produced by the model encode semantic information in a way reminiscent of word embedding approaches, making them useful for a range of related tasks such as tag recommendations.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information Filtering*

Keywords

Recommender Systems, Cold-start, Matrix Factorization

1. INTRODUCTION

Building recommender systems that perform well in cold-start scenarios (where little data is available on new users and items) remains a challenge. The standard matrix factorisation (MF) model performs poorly in that setting: it is difficult to effectively estimate user and item latent factors when collaborative interaction data is sparse.

Content-based (CB) methods address this by representing items through their metadata [10]. As these are known in advance, recommendations can be computed even for new items for which no collaborative data has been gathered. Unfortunately, no transfer learning occurs in CB models: models for each user are estimated in isolation and do not benefit from data on other users. Consequently, CB models perform worse than MF models where collaborative information is available and require a large amount of data on each user, rendering them unsuitable for user cold-start [1].

At Lyst, solving these problems is crucial. We are a fashion company aiming to provide our users with a convenient and engaging way to browse—and shop—for fashion online. To that end we maintain a very large product catalogue: at the time of writing, we aggregate over 8 million fashion items from across the web, adding tens of thousands of new products every day.

Three factors conspire to make recommendations challenging for us. Firstly, our system contains a very large number of items. This makes our data very sparse. Secondly, we deal in fashion: often, the most relevant items are those from newly released collections, allowing us only a short window to gather data and provide effective recommendations. Finally, a large proportion of our users are first-time visitors: we would like to present them with compelling recommendations even with little data. This combination of user and item cold-start makes both pure collaborative and content-based methods unsuitable for us.

To solve this problem, I use a hybrid content-collaborative model, called LightFM due to its resemblance to factorisation machines (see Section 3). In LightFM, like in a collaborative filtering model, users and items are represented as latent vectors (embeddings). However, just as in a CB model, these are entirely defined by functions (in this case, linear combinations) of embeddings of the content features that describe each product or user. For example, if the movie 'Wizard of Oz' is described by the following features: *'musical fantasy'*, *'Judy Garland'*, and *'Wizard of Oz'*, then its latent representation will be given by the sum of these features' latent representations.

In doing so, LightFM unites the advantages of content-based and collaborative recommenders. In this paper, I formalise the model and present empirical results on two datasets, showing that:

1. In both cold-start and low density scenarios, LightFM performs at least as well as pure content-based models, substantially outperforming them when either *(1)* collaborative information is available in the training set or *(2)* user features are included in the model.

2. When collaborative data is abundant (warm-start, dense user-item matrix), LightFM performs at least as well as the MF model.

3. Embeddings produced by LightFM encode important semantic information about features, and can be used for related recommendation tasks such as tag recommendations.

CBRecSys 2015, September 20, 2015, Vienna, Austria.

This has several benefits for real-world recommender systems. Because LightFM works well on both dense and sparse data, it obviates the need for building and maintaining multiple specialised machine learning models for each setting. Additionally, as it can use both user and item metadata, it has the quality of being applicable in both item and user cold-start scenarios.

To allow others to reproduce the results in this paper, I have released a Python implementation of LightFM[1], and made the source code for this paper and all the experiments available on Github[2].

2. LIGHTFM

2.1 Motivation

The structure of the LightFM model is motivated by two considerations.

1. The model must be able to learn user and item representations from interaction data: if items described as 'ball gown and 'pencil skirt' are consistently all liked by users, the model must learn that ball gowns are similar to pencil skirts.

2. The model must be able to compute recommendations for new items and users.

I fulfil the first requirement by using the latent representation approach. If ball gowns and pencil skirts are both liked by the same users, their embeddings will be close together; if ball gowns and biker jackets are never liked by the same users, their embeddings will be far apart.

Such representations allow transfer learning to occur. If the representations for ball gowns and pencil skirts are similar, we can confidently recommend ball gowns to a new user who has so far only interacted with pencil skirts.

This is over and above what pure CB models using dimensionality reduction techniques (such as latent semantic indexing, LSI) can achieve, as these only encode information given by feature co-occurrence rather than user actions. For example, suppose that all users who look at items described as aviators also look at items described as wayfarers, but the two features never describe the same item. In this case, the LSI vector for wayfarers will not be similar to the one for aviators even though collaborative information suggests it should be.

I fulfil the second requirement by representing items and users as linear combinations of their content features. Because content features are known the moment a user or item enters the system, this allows recommendations to be made straight away. The resulting structure is also easy to understand. The representation for denim jacket is simply a sum of the representation of denim and the representation of jacket; the representation for a female user from the US is a sum of the representations of US and female users.

2.2 The Model

To describe the model formally, let U be the set of users, I be the set of items, F^U be the set of user features, and F^I the set of item features. Each user interacts with a number of items, either in a favourable way (a positive interaction), or in an unfavourable way (a negative interaction). The set of all user-item interaction pairs $(u, i) \in U \times I$ is the union of both positive S^+ and negative interactions S^-.

Users and items are fully described by their features. Each user u is described by a set of features $f_u \subset F^U$. The same holds for each item i whose features are given by $f_i \subset F^I$. The features are known in advance and represent user and item metadata.

The model is parameterised in terms of d-dimensional user and item feature embeddings $\boldsymbol{e}_f^U$ and $\boldsymbol{e}_f^I$ for each feature f. Each feature is also described by a scalar bias term (b_f^U for user and b_f^I for item features).

The latent representation of user u is given by the sum of its features' latent vectors:

$$\boldsymbol{q}_u = \sum_{j \in f_u} \boldsymbol{e}_j^U$$

The same holds for item i:

$$\boldsymbol{p}_i = \sum_{j \in f_i} \boldsymbol{e}_j^I$$

The bias term for user u is given by the sum of the features' biases:

$$b_u = \sum_{j \in f_u} b_j^U$$

The same holds for item i:

$$b_i = \sum_{j \in f_i} b_j^I$$

The model's prediction for user u and item i is then given by the dot product of user and item representations, adjusted by user and item feature biases:

$$\widehat{r}_{ui} = f\left(\boldsymbol{q}_u \cdot \boldsymbol{p}_i + b_u + b_i\right) \tag{1}$$

There is a number of functions suitable for $f(\cdot)$. An identity function would work well for predicting ratings; in this paper, I am interested in predicting binary data, and so after Rendle *et al.* [16] I choose the sigmoid function

$$f(x) = \frac{1}{1 + \exp(-x)}.$$

The optimisation objective for the model consists in maximising the likelihood of the data conditional on the parameters. The likelihood is given by

$$L\left(\boldsymbol{e}^U, \boldsymbol{e}^I, \boldsymbol{b}^U, \boldsymbol{b}^I\right) = \prod_{(u,i) \in S^+} \widehat{r}_{ui} \times \prod_{(u,i) \in S^-} (1 - \widehat{r}_{ui}) \tag{2}$$

I train the model using asynchronous stochastic gradient descent [14]. I use four training threads for experiments performed in this paper. The per-parameter learning rate schedule is given by ADAGRAD [6].

2.3 Relationship to Other Models

The relationship between LightFM and the collaborative MF model is governed by the structure of the user and item feature sets. If the feature sets consist solely of indicator variables for each user and item, LightFM reduces to the standard MF model. If the feature sets also contain metadata features shared by more than one item or user, LightFM extends the MF model by letting the feature latent factors explain part of the structure of user interactions.

This is important on three counts.

[1] https://github.com/lyst/lightfm/

[2] https://github.com/lyst/lightfm-paper/

1. In most applications there will be fewer metadata features than there are users or items, either because an ontology with a fixed type/category structure is used, or because a fixed-size dictionary of most common terms is maintained when using raw textual features. This means that fewer parameters need to be estimated from limited training data, reducing the risk of overfitting and improving generalisation performance.

2. Latent vectors for indicator variables cannot be estimated for new, cold-start users or items. Representing these as combinations of metadata features that *can* be estimated from the training set makes it possible to make cold-start predictions.

3. If only indicator features are present, LightFM should perform on par with the standard MF model.

When only metadata features and no indicator variables are present, the model in general does not reduce to a pure content-based system. LightFM estimates feature embeddings by factorising the collaborative interaction matrix; this is unlike content-based systems which (when dimensionality reduction is used) factorise pure content co-occurrence matrices.

One special case where LightFM does reduce to a pure CB model is where each user is described by an indicator variable and has interacted only with one item. In that setting, the user vector is equivalent to a document vector in the LSI formulation, and only features which occur together in product descriptions will have similar embeddings.

The fact that LightFM contains both the pure CB model at the sparse data end of the spectrum and the MF model at the dense end suggests that it should adapt well to datasets of varying sparsity. In fact, empirical results show that it performs at least as well as the appropriate specialised model in each scenario.

3. RELATED WORK

There are a number of related hybrid models attempting to solve the cold-start problem by jointly modelling content and collaborative data.

Soboroff *et al.* [21] represent users as linear combinations of the feature vectors of items they have interacted with. They then perform LSI on the resulting item-feature matrix to obtain latent user profiles. Representations of new items are obtained by projecting them onto the latent feature space. The advantage of the model, relative to pure CB approaches, consists in using collaborative information encoded in the user-feature matrix. However, it models user preferences as being defined over individual features themselves instead of over items (sets of features). This is unlike LightFM, where a feature's effect in predicting an interaction is always taken in the context of all other features characterising a given user-item pair.

Saveski *et al.* [18] perform joint factorisation of the user-item and item-feature matrices by using the same item latent feature matrix in both decompositions; the parameters are optimised by minimising a weighted sum of both matrices' reproduction loss functions. A weight hyperparameter governs the relative importance of accuracy in decomposing the collaborative and content matrices. A similar approach is used by McAuley *et al.* [11] for jointly modelling ratings and product reviews. Here, LightFM has the advantage of simplicity as its single optimisation objective is to factorise the user-item matrix.

Shmueli *et al.* [20] represent items as linear combinations of their features' latent factors to recommend news articles; like LightFM, they use a single-objective approach and minimise the user-item matrix reproduction loss. They show their approach to be successful in a modified cold-start setting, where both metadata and data on other users who have commented on a given article is available. However, their approach does not extend to modelling user features and does not provide evidence on model performance in warm-start scenario.

LightFM fits into the hybrid model tradition by jointly factorising the user-item, item-feature, and user-feature matrices. From a theory standpoint, it can be construed as a special case of Factorisation Machines [15].

FMs provide an efficient method of estimating variable interaction terms in linear models under sparsity. Each variable is represented by a k-dimensional latent factor; the interaction between variable i and j is then given by the dot product of their latent factors. This has the advantage of reducing the number of parameters to be estimated.

LightFM further restricts the interaction structure by only estimating the interactions between user and item features. This aids the interpretability of resulting feature embeddings.

4. DATASETS

I evaluate LightFM's performance on two datasets. The datasets span the range of dense interaction data, where MF models can be expected to perform well (MovieLens), and sparse data, where CB models tend to perform better (CrossValidated). Both datasets are freely available.

4.1 MovieLens

The first experiment uses the well-known MovieLens 10M dataset[3], combined with the Tag Genome tag set [22].

The dataset consists of approximately 10 million movie ratings, submitted by $71,567$ users on $10,681$ movies. All movies are described by their genres and a list of tags from the Tag Genome. Each movie-tag pair is accompanied by a relevance score (between 0 and 1), denoting how accurately a given tag describes the movie.

To binarise the problem, I treat all ratings below 4.0 (out of a 1 to 5 scale) as negative; all ratings equal to or above 4.0 are positive. I also filter out all ratings that fall below the 0.8 relevance threshold to retain only highly relevant tags.

The final dataset contains $69,878$ users, $10,681$ items, $9,996,948$ interactions, and 1030 unique tags.

4.2 CrossValidated

The second dataset consists of questions and answers posted on CrossValidated[4], a part of the larger network of StackExchange collaborative Q&A sites that focuses on statistics and machine learning. The dataset[5] consists of 5953 users, $44,200$ questions, and $188,865$ answers and comments. Each question is accompanied by one or more of 1032 unique tags (such as 'regression' or 'hypothesis-testing'). Additionally,

[3] `http://grouplens.org/datasets/movielens/`

[4] `http://stats.stackexchange.com`

[5] `https://archive.org/details/stackexchange`

user metadata is available in the form of 'About Me' sections on users' profiles.

The recommendation goal is to match users with questions they can answer. A user answering a question is taken as an implicit positive signal; all questions that a user has not answered are treated as implicit negative signals. For the training and test sets, I construct 3 negative training pairs for each positive user-question pair by randomly sampling from all questions that a given user has not answered.

To keep the model simple, I focus on a user's willingness to answer a question rather than their ability, and forego modelling user expertise [17].

Table 1: Results

	CrossValidated		MovieLens	
	Warm	Cold	Warm	Cold
LSI-LR	0.662	0.660	0.686	0.690
LSI-UP	0.636	0.637	0.687	0.681
MF	0.541	0.508	0.762	0.500
LightFM (tags)	0.675	0.675	0.744	0.707
LightFM (tags + ids)	0.682	0.674	**0.763**	**0.716**
LightFM (tags + about)	**0.695**	**0.696**		

5. EXPERIMENTAL SETUP

For each dataset, I perform two experiments. The first simulates a warm-start setting: 20% of all interaction pairs are randomly assigned to the test set, but all items and users are represented in the training set. The second is an item cold-start scenario: all interactions pertaining to 20% of items are removed from the training set and added to the test set. This approximates a setting where the recommender is required to make recommendations from a pool of items for which no collaborative information has been gathered, and only content metadata (tags) are available.

I measure model accuracy using the mean receiver operating characteristics area under the curve (ROC AUC) metric. For an individual user, AUC corresponds to the probability that a randomly chosen positive item will be ranked higher than a randomly chosen negative item. A high AUC score is equivalent to low rank-inversion probability, where the recommender mistakenly ranks an unattractive item higher than an attractive item. I compute this metric for all users in the test set and average it for the final score.

I compute the AUC metric by repeatedly randomly splitting the dataset into a 80% training set and a 20% test set. The final score is given averaging across 10 repetitions.

I test the following models:

1. **MF**: a conventional matrix factorisation model with user and item biases and a sigmoid link function [8].

2. **LSI-LR**: a content-based model. To estimate it, I first derive latent topics from the item-feature matrix through latent semantic indexing and represent items as linear combinations of latent topics. I then fit a separate logistic regression (LR) model for each user in the topic mixture space. Unlike the LightFM model, which uses collaborative data to produce its latent representation, LSI-LR is purely based on factorising the content matrix. It should therefore be helpful in highlighting the benefit of using collaborative information for constructing feature embeddings.

3. **LSI-UP**: a hybrid model that represents user profiles (UP) as linear combinations of items' content vectors, then applies LSI to the resulting matrix to obtain latent user and item representations ([21], see Section 3). I estimate this model by first constructing a user-feature matrix: each row represents a user and is given by the sum of content feature vectors representing the items that user positively interacted with. I then apply truncated SVD to the normalised matrix to obtain user and feature latent vectors; item latent vectors are obtained through projecting them onto the latent feature space. The recommendations score for a user-item pair is then the inner product of their latent representations.

4. **LightFM (tags)**: the LightFM model using only tag features.

5. **LightFM (tags + ids)**: the LightFM model using both tag and item indicator features.

6. **LightFM (tags + about)**: the LightFM model using both item and user features. User features are available only for the CrossValidated dataset. I construct them by converting the 'About Me' sections of users' profiles to a bag-of-words representation. I first strip them of all HTML tags and non-alphabetical characters, then convert the resulting string to lowercase and tokenise on spaces.

In both LightFM (tags) and LightFM (tags + ids) users are described only by indicator features.

I train the LightFM models using stochastic gradient descent with an initial learning rate of 0.05. The latent dimensionality of the models is set to 64 for all models and experiments. This setting is intended to reflect the balance between model accuracy and the computational cost of larger vectors in production systems (additional results on model sensitivity to this parameter are presented in Section 6.2). I regularise the model through an early-stopping criterion: the training is stopped when the model's performance on the test set stops improving.

6. EXPERIMENTAL RESULTS

6.1 Recommendation accuracy

Experimental results are summarised in Table 1. LightFM performs very well, outperforming or matching the specialised model for each scenario.

In the **warm-start, low-sparsity** case (warm-start MovieLens), LightFM outperforms MF slightly when using both tag and item indicator features. This suggest that using metadata features may be valuable even when abundant interaction data is present.

Notably, LightFM (tags) almost matches MF performance despite using only metadata features. The LSI-LR and LSI-UP models using the same information fare much worse. This demonstrates that *(1)* it is crucial to use collaborative information when estimating content feature embeddings, and *(2)* LightFM can capture that information much more accurately than other hybrid models such as LSI-UP.

In the **warm-start, high-sparsity** case (warm-start CrossValidated), MF performs very poorly. Because user interaction data is sparse (the CrossValidated user-item matrix is 99.95% sparse vs only 99% for the MovieLens dataset), MF is unable to learn good latent representations. Content-based models such as LSI-LR perform much better.

LightFM variants provide the best performance. LightFM (tags + about) is by far the best model, showing the added advantage of LightFM's ability to integrate user metadata embeddings into the recommendation model. This is likely due to improved prediction performance for users with little data in the training set.

Results for the **cold-start** cases are broadly similar. On the CrossValidated dataset, all variants of LightFM outperform other models; LightFM (tags + about) again provides the best performance. Interestingly, LightFM (tags + indicators) outperforms LightFM (tags) slightly on the MovieLens dataset, even though no embeddings can be estimated for movies in the test set. This suggests that using both metadata and per-movie features allows the model to estimate better embeddings for both, much like the use of user and item bias terms allows better latent factors to be computed. Unsurprisingly, MF performs no better than random in the cold-start case.

In all scenarios the LSI-UP model performs no better than the LSI-LR model, despite its attempt to incorporate collaborative data. On the CrossValidated dataset it performs strictly worse. This might be because its latent representations are estimated on less data than in LSI-LR: as there are fewer users than items in the dataset, there are fewer rows in the user-feature matrix than in the item-feature matrix.

The results confirm that LightFM encompasses both the MF and the LSI-LR model as special cases, performing better than the LSI-LR model in the sparse-data scenario and better than the MF model in the dense-data case. This means not only that a single model can be maintained in either settings, but also that the model will continue to perform well even when the sparsity structure of that data changes.

Good performance of LightFM (tags) in both datasets is predicated on the availability of high-quality metadata. Nevertheless, it is often possible to obtain good quality metadata from item descriptions (genres, actor lists and so on), expert or community tagging (*Pandora* [23], *StackOverflow*), or computer vision systems where image or audio data is available (we use image-based convolutional neural networks for product tagging). In fact, the feature embeddings produced by LightFM can themselves be used to assist the tagging process by suggesting related tags.

6.2 Parameter Sensitivity

Figure 1 plots the accuracy of LightFM, LSI-LR, and LSI-UP against values of the latent dimensionality hyperparameter d in the cold-start scenario (averaged over 30 runs of each algorithm). As d increases, each model is capable of modelling more complex structures and achieves better performance.

Interestingly, LightFM performs very well even with a small number of dimensions. In both datasets LightFM consistently outperforms other models, achieving high performance with as few as 16 dimensions. On CrossValidated data, it achieves the same performance as the LSI-LR model for much smaller d: it matches the accuracy of the 512-dimensional LSI-LR model even when using fewer than 32 dimensions.

Table 2: Tag similarity

Query tag	Similar tags
'regression'	'least squares', 'multiple regression', 'regression coefficients', 'multicollinearity'
'MCMC'	'BUGS', 'Metropolis-Hastings', 'Beta-Binomial', 'Gibbs', 'Bayesian'
'survival'	'epidemiology', 'Cox model', 'Kaplan-Meier', 'hazard'
'art house'	'pretentious', 'boring', 'graphic novel', 'pointless', 'weird'
'dystopia'	'post-apocalyptic', 'futuristic', 'artificial intelligence'
'bond'	'007', 'secret service', 'nuclear bomb', 'spying', 'assassin'

This is an important win for large-scale recommender systems, where the choice of d is governed by a trade-off between vector size and recommendation accuracy. Since smaller vectors occupy less memory and use fewer computations during query time, better representational power at small d allows the system to achieve the same model performance at a smaller computational cost.

6.3 Tag embeddings

Feature embeddings generated by the LightFM model capture important information about the semantic relationships between different features. Table 2 gives some examples by listing groups of tags similar (in the cosine similarity sense) to a given query tag.

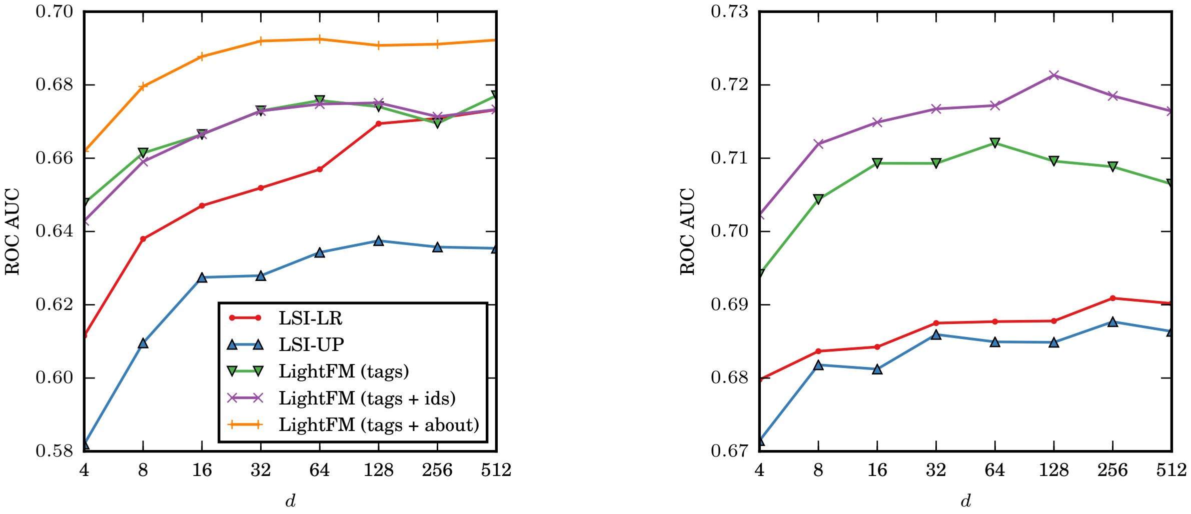
In this respect, LightFM is similar to recent word embedding approaches like word2vec and GloVe [12, 13]. This is perhaps unsurprising, given that word embedding techniques are closely related to forms of matrix factorisation [9]. Nevertheless, LightFM and word embeddings differ in one important respect: whilst word2vec and GloVe embeddings are driven by textual corpus co-incidence statistics, LightFM is based on user interaction data.

LightFM embeddings are useful for a number of recommendation tasks.

1. **Tag recommendation.** Various applications use collaborative tagging as a way of generating richer metadata for use in search and recommender system [2, 7]. A tag recommender can enhance this process by either automatically applying matching tags, or generating suggested tags lists for approval by users. LightFM-produced tag embeddings will work well for this task without the need to build a separate specialised model for tag recommendations.

2. **Genre or category recommendation.** Many domains are characterised by an ontology of genres or categories which play an important role in the presentation of recommendations. For example, the Netflix interface is organised in genre rows; for Lyst, fashion designers, categories and subcategories are fundamental. The degree of similarity between the embeddings of genres or categories provides a ready basis for genre or category recommendations that respect the semantic structure of the ontology.

Figure 1: Latent dimension sensitivity



(a) CrossValidated

(b) MovieLens

3. **Recommendation justification.** Rich information encoded in feature embeddings can help provide explanations for recommendations made by the system. For example, we might recommend a ball gown to a user who likes pencil skirts, and justify it by the two features' similarity as revealed by the distance between their latent factors.

7. USAGE IN PRODUCTION SYSTEMS

The LightFM approach is motivated by our experience at Lyst. We have deployed LightFM in production, and successfully use it for a number of recommendation tasks. In this section, I describe some of the engineering and algorithm choices that make this possible.

7.1 Model training and fold-in

Thousands of new items and users appear on Lyst every day. To cope with this, we train our LightFM model in an online manner, continually updating the representations of existing features and creating fresh representations for features that we have never observed before.

We store model state, including feature embeddings and accumulated squared gradient information in a database. When new data on user interaction arrives, we restore the model state and resume training, folding in any newly observed features. Since our implementation uses per-parameter diminishing learning rates (ADAGRAD), any updates of established features will be incremental as the model adapts to new data. For new features, a high learning rate is used to allow useful embeddings to be learned as quickly as possible.

No re-training is necessary for folding in new products: their representation can be immediately computed as the sum of the representations of their features.

7.2 Feature engineering

Each of our products is described by a set of textual features as well as structured metadata such as its type (dress, shoes and so on) or designer. These are accompanied by additional features coming from two sources.

Firstly, we employ a team of experienced fashion moderators, helping us to derive more fine-grained features such as clothing categories and subcategories (peplum dress, halterneck and so on).

Secondly, we use machine learning systems for automatic feature detection. The most important of these is a set of deep convolutional neural networks deriving feature tags from product image data.

7.3 Approximate nearest neighbour searches

The biggest application of LightFM-derived item representations are related product recommendations: given a product, we would like to recommend other highly relevant products. To do this efficiently across 8 million products, we use a combination of approximate (for on-demand recommendations) and exact (for near-line computation) nearest neighbour search.

For approximate nearest neighbour (ANN) queries, we use Random Projection (RP) trees [4, 5]. RP trees are a variant of random-projection [3] based locality sensitive hashing (LSH).

In LSH, k-bit hash codes for each point $\boldsymbol{x}$ are generated by drawing random hyperplanes $\boldsymbol{v}$, and then setting the k-th bit of the hash code to 1 if $\boldsymbol{x} \cdot \boldsymbol{v} \geq 0$ and 0 otherwise. The approximate nearest neighbours of $\boldsymbol{x}$ are then other points that share the same hash code (or whose hash codes are within some small Hamming distance of each other).

While extremely fast, LSH has the undesirable property of sometimes producing very highly unbalanced distribution of points across all hash codes: if points are densely concentrated, many codes of the tree will apply to no products while some will describe a very large number of points. This is unacceptable when building a production system, as it will lead to many queries being very slow.

RP trees provide much better guarantees about the size of leaf nodes: at each internal node, points are split based on the median distance to the chosen random hyperplane. This guarantees that at every split approximately half the points will be allocated to each leaf, making the distribution of points (and query performance) much more predictable.

8. CONCLUSIONS AND FUTURE WORK

In this paper, I have presented an effective hybrid recommender model dubbed LightFM. I have shown the following:

1. LightFM performs at least as well as a specialised model across a wide range of collaborative data sparsity scenarios. It outperforms existing content-based and hybrid models in cold-start scenarios where collaborative data is abundant or where user metadata is available.

2. It produces high-quality content feature embeddings that capture important semantic information about the problem domain, and can be used for related tasks such as tag recommendations.

Both properties make LightFM an attractive model, applicable both in cold- and warm-start settings. Nevertheless, I see two promising directions in extending the current approach.

Firstly, the model can be easily extended to use more sophisticated training methodologies. For example, an optimisation scheme using Weighted Approximate-Rank Pairwise loss [24] or directly optimising mean reciprocal rank could be used [19].

Secondly, there is no easy way of incorporating visual or audio features in the present formulation of LightFM. At Lyst, we use a two-step process to address this: we first use convolutional neural networks (CNNs) on image data to generate binary tags for all products, and then use the tags for generating recommendations. We conjecture that substantial improvements could be realised if the CNNs were trained with recommendation loss directly.

9. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. 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*, 17(6):734–749, 2005.

[2] M. Bastian, M. Hayes, W. Vaughan, S. Shah, P. Skomoroch, H. Kim, S. Uryasev, and C. Lloyd. Linkedin skills: large-scale topic extraction and inference. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 1–8. ACM, 2014.

[3] S. Dasgupta. Experiments with random projection. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 143–151. Morgan Kaufmann Publishers Inc., 2000.

[4] S. Dasgupta and Y. Freund. Random projection trees and low dimensional manifolds. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 537–546. ACM, 2008.

[5] S. Dasgupta and K. Sinha. Randomized partition trees for exact nearest neighbor search. *arXiv preprint arXiv:1302.1948*, 2013.

[6] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

[7] R. Jäschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme. Tag recommendations in folksonomies. In *Knowledge Discovery in Databases: PKDD 2007*, pages 506–514. Springer, 2007.

[8] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[9] O. Levy and Y. Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185, 2014.

[10] P. Lops, M. De Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

[11] J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM, 2013.

[12] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[13] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12, 2014.

[14] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 693–701, 2011.

[15] S. Rendle. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 995–1000. IEEE, 2010.

[16] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461. AUAI Press, 2009.

[17] J. San Pedro and A. Karatzoglou. Question recommendation for collaborative question answering systems with RankSLDA. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 193–200. ACM, 2014.

[18] M. Saveski and A. Mantrach. Item cold-start recommendations: learning local collective embeddings. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 89–96. ACM, 2014.

[19] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic. CLiMF: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the 6th ACM onference on Recommender systems*, pages 139–146. ACM, 2012.

[20] E. Shmueli, A. Kagian, Y. Koren, and R. Lempel. Care to Comment?: Recommendations for commenting on news stories. In *Proceedings of the 21st international conference on World Wide Web*, pages 429–438. ACM, 2012.

[21] I. Soboroff and C. Nicholas. Combining content and collaboration in text filtering. In *Proceedings of the IJCAI*, volume 99, pages 86–91, 1999.

[22] J. Vig, S. Sen, and J. Riedl. The tag genome: Encoding community knowledge to support novel interaction. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 2, 2012.

[23] T. Westergren. The music genome project. *Online: http://pandora. com/mgp*, 2007.

[24] J. Weston, S. Bengio, and N. Usunier. WSABIE: Scaling up to large vocabulary image annotation. In *IJCAI*, volume 11, pages 2764–2770, 2011.

Conceptual Impact-Based Recommender System for CiteSeer^X

Kevin Labille
Department of Computer Science and Computer Engineering
University of Arkansas
Fayetteville, AR 72701, USA
kclabill@uark.edu

Susan Gauch
Department of Computer Science and Computer Engineering
University of Arkansas
Fayetteville, AR 72701, USA
sgauch@uark.edu

Ann Smittu Joseph
Department of Computer Science and Computer Engineering
University of Arkansas
Fayetteville, AR 72701, USA
ann@email.uark.edu

ABSTRACT

CiteSeer^x is a digital library for scientific publications written by Computer Science researchers. Users are able to retrieve relevant documents from the database by searching by author name and/or keyword queries. Users may also receive recommendations of papers they might want to read provided by an existing conceptual recommender system. This system recommends documents based on an automatically-constructed user profile. Unlike traditional content-based recommender systems, the documents and the user profile are represented as concepts vectors rather than keyword vectors and papers are recommended based on conceptual matches rather than keyword matches between the profile and the documents. Although the current system provides recommendations that are on-topic, they are not necessarily high quality papers. In this work, we introduce the Conceptual Impact-Based Recommender (CIBR), a hybrid recommender system that extends the existing conceptual recommender system in CiteSeer^x by including an explicit quality factor as part of the recommendation criteria. To measure quality, our system considers the impact factor of each paper's authors as measured by the authors' h-index. Experiments to evaluate the effectiveness of our hybrid system show that the CIBR system recommends more relevant papers as compared to the conceptual recommender system.

Categories and Subject Descriptors

Information Systems [**Information retrieval**]: **Retrieval tasks and goals**:Recommender systems

General Terms

Performance, Reliability, Design, Experimentation

Keywords

Recommender System, h-index, Content-based Recommender System, CiteSeer^x, Information Retrieval

CBRecSys 2015, September 20, 2015, Vienna, Austria.

1. INTRODUCTION

In recent years, recommender systems have become ubiquitous, recommending movies, restaurants, and books etc. The recommendations ease information overload for users by pro-actively suggesting relevant items to the users, moving the burden of discovery from the user to the system. The number and type of applications that use recommender systems keeps growing [1]; one practical application that is of interest to researchers in any domain is the ability of recommender systems to suggest relevant scientific literature. These systems can expedite scientific innovation by helping researchers keep abreast of new publications in their fields and also help new researchers learn about the most important literature in an area new to them. Digital libraries can employ recommender systems that suggest papers to their users based on each user's research interests. However, an effective recommender system should not only consider the subject of a paper, it should also take into account the paper's quality when making recommendations. To this end, we present a recommender system that recommends scientific papers based on user preferences as well as paper quality as measured by the authors' impact factors to provide recommendations of high-quality papers that are relevant to the user's research area. To help CiteSeer^x users locate scientific papers related to their work, a citation-based recommender system was developed by Chandrasekaran et al. in 2008 [4] . Although citations are effective at identifying papers that have relevant content and are also high quality, this approach is only effective in recommending papers with many citations. These unfortunately tend to be older papers that have been published long enough ago to generate many citations. Especially in a fast-moving domain like computer science, researchers need to know about recent contributions to their field, yet recent papers have few citations. To solve this problem, a content-based recommender system for CiteSeer^x was developed by Pudhiyaveetil et al.[8]. This conceptual recommender system automatically builds conceptual profiles for users based on their interactions with the system. It also builds conceptual profiles for each document and recommends papers based on conceptual matches between document and user profiles. Even though the recommendations were shown to be more relevant than those produced by a keyword-based recommender system, they are not always high quality papers that the researcher wanted to read. Our objective is to improve upon the conceptual recommender system by providing better quality recommendations to the users. To do so, we developed a recommender system that recommends papers based on the paper authors' impact factors. We combined the impact-factor based recommendations with the concept-based recommendations in varying proportions to create a hybrid recommender system. We evaluated the effectiveness of the conceptual recommender system, the impact-factor recommender system, and the hybrid recommender system and found that the hybrid recommender system provides the most accurate recommendations. The rest of this paper is organized as follows: In section 2 we review related work. Section 3 describes the Conceptual Impact-Based Recommender (CBIR) system in detail. In section 4, we present our experimental evaluation to analyze the effectiveness of our recommender system. Finally, we present our conclusions and discuss future work in section 5.

2. RELATED WORK

The design of a recommender system can vary based on the nature of user feedback or the availability of data. There are three main approaches: collaborative filtering, content based recommender systems, and recommender systems that are a hybrid of the two [1]. The first approach generates recommendations based on similarities between the users' behavior or/and preferences. In contrast, content-based approaches recommend items to the users based on similarities between the attributes of the items themselves [10]. Collaborative approaches are typically used when semantic features cannot easily be extracted from the items, so indirect evidence based on user's likes or ratings must be compared. To be effective, collaborative filtering requires a large active user community to avoid the well-known "cold-start" problem in which there are many more items to be recommended than there are users with likes or ratings upon which recommendations can be based. On the other hand, pure content-based recommender systems do not consider external information that might be available from the users, e.g., popularity. For these reasons, many recommender systems employ a hybrid approach combines both of the previously-described approaches.

Content-based recommender systems match the users' preferences to each items' features to recommend new objects [10]. Many share the approach of building a user profile from a set of features extracted from previously liked items. This user profile is then compared to the features of all items in the collection and the most similar items are recommended to the user [12]. This type of recommender system can be used in domain for which semantically relevant features can be extracted and it is particularly well-suited for domains that include textual items as scientific literature or domains with annotations such as movies or music [12]. Kompan et al. used this approach to recommend news articles on a web site [9]. In this domain, the volume of articles and the dynamic nature of news make collaborative filtering infeasible so they implemented a content-based recommender system based on cosine similarity that suggested articles that best matched an implicitly constructed user model [9].

Our work is a hybrid approach that enhances a content-based recommender system with a quality measure to recommend scientific literature. According to Beel et al., recommender systems for research papers are flourishing with more than 80 approaches existing today that have been discussed in over 170 articles and patents [2]. Such recommender systems are useful for researchers to be up to date in their research area. Many content-based recommender systems represent the user interests and the documents as weighted keyword vectors. One example is [13] in which $tf * idf$ weights are calculated for keywords and the cosine similarity measure is used to determine the relevancy of a paper to a user's profile. An approach similar to ours is used in [5]. In their work, each paper's features are represented as concepts created by automatically extracting keyphrases. User profiles are constructed from the concepts in previously viewed papers and the recommender system matches the user profile concepts to each papers' concepts to suggest new papers in a scientific library. In [8], a conceptual recommender system was presented that recommends research papers for CiteSeer^x users. Unlike the previous work, the concepts for each paper are assigned by automatically classifying papers into a set of concepts defined by a pre-existing ontology. A conceptual user profile is implicitly built as users view papers in the collection and this user profile is used to recommend conceptually similar papers.

The content-based recommender systems can recommend literature that is similar in topic to the user's profile, but it does not necessarily recommend high-quality papers. Although there is no perfect way to measure the quality of articles, the Impact Factor (IF) introduced in 1955 is still considered the best way to evaluate a paper's scientific merit [6]. There are several types of IFs, including the widely used h-index that evaluates a researcher's impact [7]. It has been recently used is several fields such as health services research [3], business and management [11] or even academic psychiatry [14] . Although the work in [5], [8], and [13] are similar to ours, our recommender system expands upon their work by incorporating a quality factor as measured by the authors' h-indexes.

3. APPROACH

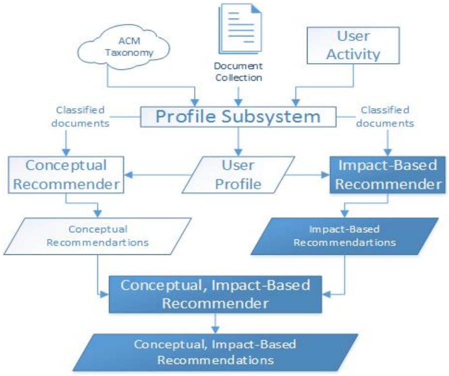


Figure 1: Architecture of the CIBR

The architecture of the Conceptual Impact-Based Recommender System (CIBR) is shown in Figure 1. The Profile Subsystem classifies all documents in the CiteSeer^x database into the 369 predefined categories in the ACM Computing Classification System (CCS). Documents manually tagged with ACM categories by their authors are used as the training set for a k-nearest neighbor classifier. As users interact with the system, the documents that they examine are input to the Profile Subsystem. The categories associated with each examined document are combined to create a weighted conceptual user profile. This user profile is used by both the Conceptual Recommender and the Impact-Based Recommender described in the following sections. The outputs of these two Recommenders are combined to produce the recommendations from the CBIR.

3.1 Concept-Based Recommender System

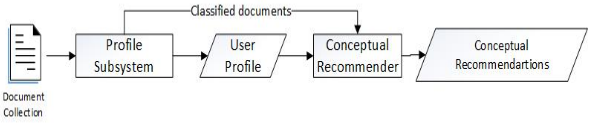


Figure 2: Conceptual Recommender System Architecture

As a user views documents in CiteSeer^x, the Profile Subsystem builds a conceptual user profile for them by accumulating the concept weights associated with the documents that the user examines. The Conceptual Recommender System then recommends documents to the user based on the similarity between each document's conceptual profile and the user's conceptual profile [8]. The weight of the conceptual match between document i and user j is calculated using the cosine similarity function over all M=369 concepts in the ACM taxonomy:

$$ConceptualWeight_{ij} = \sum_{K=1}^{M} (cwt_{ik} * cwt_{jk})$$

Where
cwt_{ik} = weight of concept k in document profile i and
cwt_{jk} = weight of concept k in user profile j as explained and detailed in [8].

3.2 Impact-Based Recommender System

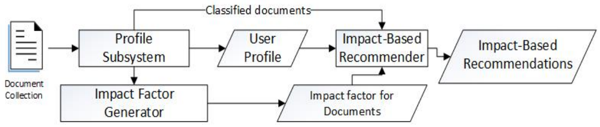


Figure 3: Impact-based Recommender System Architecture

The Impact Factor Generator precalculates an impact factor for each document in the collection as measured by its authors' h-indices. As described by Hirsch, an author has an h-index of m based on his/her N published articles if m articles have at least m citations each, and the other N-m articles have no more than m citations each [7]. The impact factor for a document is calculated by finding the h-index value of each of the authors of the document and then selecting the highest h-index value. Thus, document i's h-index is equal that of its most impactful author:

$$ImpactWeight_i = \max_{l \in A_{il}}(hindex_{il}) \qquad (1)$$

Where
A_{il} = list of the authors l of document i
Since the impact factor is independent of users, the Impact-Based recommendations would be the same for all users, i.e., the most impactful documents in the entire collection. We do, however, use the user profile to filter out documents from categories in which the user has shown no previous input. Thus, Impact-Based Recommender returns high-impact documents from categories of some interest to the user. We tried other approaches to calculate the impact factor among which we consider the sum of each authors' h-indices. This particular method is limited since the highest weighted papers would usually be the ones with many authors.

3.3 Conceptual Impact-Based Recommender System

The Conceptual Impact-Based Recommender System (CIBR) combines the Conceptual Weights and the Impact Weights to produce its recommendations. The two sub-component weights are normalized to fall between 0 to 1 using linear scaling and then combined based on a tunable parameter, α. The weight of the conceptual impact match between document i and user j, γ_{ij}, is calculated using:

$$\gamma_{ij} = \alpha * C'_{ij} + (1 - \alpha) * I'_i \qquad (2)$$

Where

$$C'_{ij} = \text{normalized } ConceptualWeight_{ij} = \frac{ConceptualWeight_{ij} - min_j(ConceptualWeight)}{max_j(ConceptualWeight) - min_j(ConceptualWeight)}$$

$$I'_i = \text{normalized } ImpactWeight_i = \frac{ImpactWeight_i - min_j(ImpactWeight)}{max_j(ImpactWeight) - min_j(ImpactWeight)}$$

α = controls the relative contributions of two sub-weights

By varying α from 0 to 1, we can adjust the relative contributions of two underlying recommender systems. When $\alpha = 0$, the CBIR is a pure impact-based recommender system whilst when $\alpha = 1$, the CBIR is a purely Conceptual recommender system.

4. EXPERIMENTAL EVALUATION

4.1 Subjects and Dataset

We conducted several experiments to measure the effectiveness of our hybrid recommender system. Experiments were done with 30 subjects, undergraduate and graduate computer science and computer engineering students from the university of Arkansas. We use the 2190179 documents in our snapshot of the CiteSeer^x, a digital library and a search engine for computer and information sciences literature. Because previous experiments have shown that profiles become stable after viewing 20 papers, users we asked to search for and view at least that many papers related to their own research area. Based on those documents, user profiles were automatically constructed for each user

4.2 Evaluation Method

The goal of this experiment was first to determine what combination the conceptual match and the paper quality is most effective in our hybrid recommender system. The relative combinations of the two is given by the equation in Section 3. By changing the value of α we are able to control the relative contributions of the two recommender systems with α = 0.0 being a pure impact-based recommender system and α = 1.0 being a pure conceptual recommender system and α = 0.5 using even contributions from both. We varied the value of α from 0.0 to 1.0 with an increment of 0.1 for each of the subjects in the experiment and for each value of α we collected the top ten recommended documents. For each

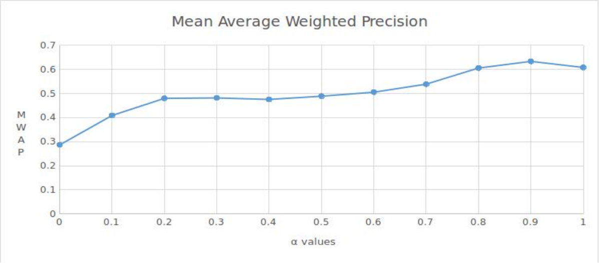


Figure 4: Mean Average Weighted Precision for every α

user, we presented them with the set of all documents recommended by any of the versions of the system (removing duplicates) in random order. They provided explicit relevance feedback by rating the papers as very relevant (2), relevant (1), or irrelevant (0). We then used the Mean Average Weighted Precision (MAWP) of each user for each α as a metric. The MAWP is essentially the Mean Average Precision modified to handle weights from 0..2 rather than just Boolean relevance judgments. The mean of every MAWP for each α is calculated and summarized in Figure 4. As shown on Figure 4, an α of 0.9 gives the best results, 0.6355, meaning that a 90% contribution from the conceptual recommender system and a 10% contribution from the impact-based recommender performed the best. For the second part of our analysis, we compared the effectiveness of the three recommender systems head-to-head. The hybrid recommender system with $\alpha = 0.9$ outperformed the conceptual recommender system's MWAP of 0.6083 ($\alpha = 1.0$) by 4.5% relative (or 2.72% absolute) and the impact-based recommender system's MWAP of 0.2867 ($\alpha = 0.0$) by 121.67% relative or 34.88% absolute. Both of these results are statistically significant ($p < 0.05$), based on the paired two-tailed student t-test.

5. CONCLUSION AND FUTURE WORK

In this paper, a hybrid recommender system was introduced that recommends high quality papers to CiteSeer^x users. The new recommender combines a conceptual recommender system along with an impact-factor-based recommender system. The former incorporates the user's preferences represented as a concept vector whilst the latter incorporates paper quality using the authors' impact factors as measured by their h-indexes. User experiments were conducted to compare the concept-based recommender system and the impact-based recommender system with our hybrid system. The results confirm that our hybrid recommender generates relevant documents as compared to the conceptual or the impact-factor-based recommender. Future work could consider using social networks of co-authors or differential weighting of the papers. Another direction would be to investigate the effectiveness of our hybrid recommender system by considering the g-index that gives a stronger weight to highly-cited papers as compared to the h-index. Alternatively, we could use the e-index that complements the h-index by distinguishing authors having the same h-index but different numbers of citations.

6. ACKNOWLEDGMENTS

This research was supported in part by the National Science Foundation grant number 0958123 : Collaborative Research: CI-ADDO-EN: Semantic CiteSeer^x

7. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. 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*, 17(6):734–749, 2005.

[2] J. Beel, S. Langer, M. Genzmehr, B. Gipp, C. Breitinger, and A. Nürnberger. Research paper recommender system evaluation: A quantitative literature survey. In *Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation*, pages 15–22. ACM, 2013.

[3] Y. Birks, C. Fairhurst, K. Bloor, M. Campbell, W. Baird, and D. Torgerson. Use of the h-index to measure the quality of the output of health services researchers. *Journal of health services research & policy*, 19(2):102–109, 2014.

[4] K. Chandrasekaran, S. Gauch, P. Lakkaraju, and H. P. Luong. Concept-based document recommendations for citeseer authors. In *Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 83–92. Springer, 2008.

[5] D. De Nart and C. Tasso. A personalized concept-driven recommender system for scientific libraries. *Procedia Computer Science*, 38:84–91, 2014.

[6] E. Garfield. Journal impact factor: a brief review. *Canadian Medical Association Journal*, 161(8):979–980, 1999.

[7] J. E. Hirsch. An index to quantify an individual's scientific research output. *Proceedings of the National academy of Sciences of the United States of America*, 102(46):16569–16572, 2005.

[8] A. Kodakateri Pudhiyaveetil, S. Gauch, H. Luong, and J. Eno. Conceptual recommender system for citeseerx. In *Proceedings of the third ACM conference on Recommender systems*, pages 241–244. ACM, 2009.

[9] M. Kompan and M. Bieliková. Content-based news recommendation. In *E-commerce and web technologies*, pages 61–72. Springer, 2010.

[10] P. Lops, M. De Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

[11] J. Mingers, F. Macri, and D. Petrovici. Using the h-index to measure the quality of journals in the field of business and management. *Information Processing & Management*, 48(2):234–241, 2012.

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

[13] S. Philip and A. O. John. Application of content-based approach in research paper recommendation system for a digital library. *International Journal of Advanced Computer Science & Applications*, 5(10), 2014.

[14] S. Selek and A. Saleh. Use of h index and g index for american academic psychiatry. *Scientometrics*, 99(2):541–548, 2014.

Exploiting Regression Trees as User Models for Intent-Aware Multi-attribute Diversity

Paolo Tomeo[1], Tommaso Di Noia[1], Marco de Gemmis[2], Pasquale Lops[2],
Giovanni Semeraro[2], Eugenio Di Sciascio[1]
[1] Polytechnic University of Bari – Via Orabona, 4 – 70125 Bari, Italy
[2] University of Bari Aldo Moro – Via Orabona, 4 – 70125 Bari, Italy
[1]{firstname.lastname}@poliba.it [2]{firstname.lastname}@uniba.it

ABSTRACT

Diversity in a recommendation list has been recognized as one of the key factors to increase user's satisfaction when interacting with a recommender system. Analogously to the modelling and exploitation of query intent in Information Retrieval adopted to improve diversity in search results, in this paper we focus on eliciting and using the profile of a user which is in turn exploited to represent her intents. The model is based on regression trees and is used to improve personalized diversification of the recommendation list in a multi-attribute setting. We tested the proposed approach and showed its effectiveness in two different domains, i.e. books and movies.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval

Keywords

Personalized diversity; Intent-aware diversification; Regression Trees

1. INTRODUCTION

In the recent years, diversification has gained more and more importance in the field of recommender systems. Engines able to get excellent results in terms of accuracy of results have been proved to be not effective when we consider other factors related to the quality of user experience [10]. As a matter of fact, when interacting with a system exposing a recommendation service, the user perceives as good suggestions those showing also an appropriate degree of diversity, novelty or serendipity, just to cite a few. The attitude of populating the recommendation list with similar items could exacerbate the over-specialization problem that content-based recommender systems tend to suffer from [9], even though it appears also in collaborative-filtering approaches. Improving diversity is generally a good choice to foster the user satisfaction as it increases the odds of finding relevant recommendations [1].

Here our focus is on both the *individual* (or *intra-list*) diversity, namely the degree of dissimilarity among all items in the list provided to a user, and the *aggregate diversity* [3], namely the number and distribution of distinct items recommended across all users. The item-to-item dissimilarity can be evaluated by using content-based attributes (e.g. genre in movie and music domains, product category in e-commerce) [18] or statistical information (e.g. number of co-ratings) [23]. Usually, approaches to the diversification take into account only one single attribute while, in the approach we present here, multiple attributes are selected to describe the items. The rationale behind this choice is that we believe there are numerous and heterogeneous item dimensions conditioning user's interests and choices. Moreover, depending on the user these dimensions may interact with each other thus contributing to the creation of her intents. The question is how to tackle multiple attributes to address the diversification problem.

In this paper we use regression trees as user modeling technique to infer the individual interests, useful to provide an intent-aware diversification. Compared to approaches where item attributes are treated independently one to each other, regression trees make possible to represent user tastes as a combination of interrelated characteristics. For instance, a user could have a preference for horror movies of the 80s irrespective of the director, or for horror movies of the 90s directed by a a specific director. In a regression tree, conditional probability lets to build such inference rules about user's preferences. We conducted experiments on the movie and on the book domains to empirically evaluate our approach. The performance was measured in terms of accuracy and both individual and aggregate diversity.

The main contributions of this paper are:

- a novel intent-aware diversification approach able to combine multiple attributes. It bases on the use of regression trees (and rules) to infer and encode the model of users' interests;
- a novel method to combine different diversification approaches;
- an experimental evaluation which shows the performance of the proposed approaches with respect to both accuracy and diversity measures.

The paper is organized as follows. Section 2 describes the greedy approach to diversification problem, the xQuAD algorithm and some evaluation metrics. We then continue in

Section 3 by showing how to face the multi-attribute diversification and how to leverage regression trees in the diversification process with xQuAD to provide more personalized recommendations. Section 4 describes the experimental configuration and the datasets used for the experiments while Section 5 presents and describes the experimental results, showing the competitive performance of the proposed approach. In Section 6 we review the related work at the best of our knowledge. Conclusions close the paper.

2. DIVERSITY IN RECOMMENDATIONS

The recommendation step can be followed by a re-ranking phase finalized to improve other qualities besides accuracy [3]. Some of re-ranking approaches proposed so far are based on greedy algorithms designed to handle the balance between accuracy and diversity in a recommendations list [26]. Their scheme of work is explained through Algorithm 1, where $\mathbf{P} = \langle 1, ..., n\rangle$ is the recommendation list for user u generated using the predicted ratings and the output is the re-ranked list $\mathbf{S}$ of recommendations, such that $\mathbf{S} \subset \mathbf{P}$ and whose length is $N \leq n$.

```
Data: The original recommendation list $\mathbf{P}$, $N \leq n$
Result: The re-ranked recommendation list $\mathbf{S}$
1 $\mathbf{S} = \langle\rangle$;
2 while $|\mathbf{S}| \leq N$ do
3     $i^* = \underset{i \in \mathbf{P} \setminus \mathbf{S}}{\text{argmax}}\ f_{obj}(i, \mathbf{S})$;
4     $\mathbf{S} = \mathbf{S} \circ i^*$;
5     $\mathbf{P} = \mathbf{P} \setminus \{i^*\}$
6 end
7 return $\mathbf{S}$.
```

Algorithm 1: The greedy strategy

At each iteration, the algorithm selects the item maximizing the objective function f_{obj} (line 3) – which in turn can be defined to deal with the trade-off between accuracy and diversity – and then adds it to the re-ranked list (line 4).

For our purpose, we focus on the intent-aware approach xQuAD (eXplicit Query Aspect Diversification), with the aim to diversify the user intents. It was proposed for search diversification in information retrieval by Santos et al. [15], as a probabilistic framework to explicitly model an ambiguous query as a set of sub-queries that will cover the potential aspects of the initial query. Then it was adapted for recommendation diversification by Vargas and Castells [20], replacing query and relative aspects with user and items categories, respectively. Hereafter we refer to generic *item features* - such as categories - as *features*, considering the features as possible instances of a generic attribute.

More formally, xQuAD greedily selects diverse recommendations maximizing the following objective function:

$$f_{obj}(i, \mathbf{S}, u) = \lambda\ r^*(u, i)\ +\ (1 - \lambda) div(i, \mathbf{S}, u) \qquad (1)$$

with $r^*(u, i)$ being the score predicted by the baseline recommender; the λ parameter allowing to manage the accuracy-diversity balance, where higher values give more weight to accuracy, while lower values give more weight to diversity. The last component in Equation 1 promotes the diversity, providing a measure of *novelty* with respect to the items already selected in S. As for the function $div(i, \mathbf{S}, u)$, the original formulation in [20] is:

$$div^{orig}(i, \mathbf{S}, u) = \sum_f p(i|f)p(f|u) \prod_{s \in \mathbf{S}} (1 - p(s|f)) \qquad (2)$$

where *p(i|f)* represents the likelihood of item i being chosen given the feature f while *p(f|u)* represents the user interest in the feature.

A number of measures have been proposed to evaluate the diversity in a recommendation list. Smyth and McClave [17] proposed the ILD (Intra-List Diversity), that computes the average distance between each couple of items in the list L:

$$ILD(L) = \frac{1}{|L|\,(|L| - 1)} \sum_{i,j \in L, i \neq j} (1 - sim(i, j)) \qquad (3)$$

The *sim* function is a configurable and application-dependent component which can use content-based item features or statistical information (e.g. number of co-ratings) to compute the similarity between items. We used also the metric α-nDCG, that is the redundancy-aware variant of Normalized Discounted Cumulative Gain proposed in [5]. We adopt the adapted version for recommendation proposed in [16]:

$$\alpha\text{-nDCG}(L, u) = \frac{1}{\alpha\text{-iDCG}} \sum_{r=1}^{|L|} \frac{\sum_{f \in F(L_r)} (1 - \alpha)^{cov(L, f, r-1)}}{log_2(1 + r)} \qquad (4)$$

where $cov(L, f, r - 1)$ is the number of items ranked up to position $r - 1$ containing the feature f. $F(L_r)$ represents the set of features of the r-th item. The α parameter is used to balance the emphasis between relevance and diversity. α-iDCG denotes the value of α-nDCG for the best "ideally" diversified list. Considering that the computation of the ideal value is NP-complete [5], we adopt a greedy approach: at each step we select solely the item with the highest value, regardless of the next steps.

3. INTENT-AWARE MULTI-ATTRIBUTE DIVERSITY

In this section we show how we address the intent-aware diversity problem when dealing with multi-attribute item descriptions. The presentation relies on content-based attributes (e.g. genres, years, etc. in the movies domain), but the proposed approach can be used independently of the attributes types. Therefore, one could also use statistical information as item attributes, e.g., popularity or rating variance. As explained in the previous section, we refer to features as possible instances of a generic attribute. We tried different reformulations of the *div* function in xQuAD (Equation 2) to deal with multi-attribute values. After an empirical evaluation, we chose the best div^{ma} (for **m**ulti-**a**ttribute) in terms of accuracy-diversity balance:

$$div^{ma}(i, \mathbf{S}, u) = \sum_{A \in \mathcal{A}} \frac{\sum_{f \in dom(A)} p(i|f)p(f|u)(1 - \text{avg}_{j \in \mathbf{S}}\, p(j|f))}{\sum_{f \in dom(A)} p(f|u)} \qquad (5)$$

where:

- $\mathcal{A}$ is the set of attributes;
- for each attribute $A \in \mathcal{A}$ and each feature in the attribute domain $f \in dom(A)$, $p(i|f)$ represents the importance of f for the item i. It is computed as a binary function that returns 1 if the item contains f, 0 otherwise;
- $p(f|u)$ represents the importance of the feature f for the user u and is computed as the relative frequency of the feature f on the rated items from the user u.

Here after we will refer to xQuAD using Equation 5 as *basic xQuAD*.

Besides dealing with multi-attribute descriptions, the idea behind our approach is to infer and model the user profile by means of a regression tree, a predictive model where the user interest represents the target variable, which can take continuous values. Once a regression tree is produced for a user u, then it is converted into a set of rules $RT(u)$. Each rule maps the presence/absence of a categorical feature or a constraint on a numerical one to a value v in a continuous interval. This latter indicates the predicted interest of the user on the items satisfying the rule. In our implementation we used the interval $[1, 5]$ since the value of the target variable has been calculated as the rating mean of the training instances classified by the inferred rule. Please note that the choice of a specific value interval for the target variable does not affect the overall approach. Each rule m has then the form

$$body(m) \mapsto interest = v$$

with $body(m) = \{c_1, \ldots, c_n\}$. An example of a set of rules produced for a user is shown in Figure 1.

1. $\{horror \in dom(genres),\ western \notin dom(genres),$ $DarioArgento \in dom(directors)\} \mapsto interest = 4.2$
2. $\{horror \notin dom(genres),\ thriller \in dom(genres)\}$ $\mapsto interest = 2.1$
3. $\{year > 1990,\ horror \notin dom(genres),$ $drama \in dom(genres),\ Aronofsky \in dom(directors)\}$ $\mapsto interest = 4.0$
4. $\{year < 1990,\ drama \in dom(genres),$ $AlPacino \in dom(actors)\} \mapsto interest = 3.9$
5. $\{horror \notin dom(genres)\} \mapsto interest = 3.2$

Figure 1: Example of a set of rules generated via the regression tree

Eventually, under the assumption that they represent specific user interests, the computed rules are used in the re-ranking phase as item features to improve the intent-aware recommendation diversity.

We propose also a *div* function for xQuAD so that each item is evaluated according to the rules it satisfies.

$$div^{rules}(i, \mathbf{S}, u) = \sum_{m \in M(u,i)} p(m|u)(1 - \text{avg}_{j \in \mathbf{S}}\, p(j|m)) \quad (6)$$

Here $M(u, i)$ represents the set of rules for the user u matched by the item i while $p(m|u)$ represents the importance of the rule m for u and is computed as:

$$p(m|u) = \frac{interest_m}{|M(u, i)|} \quad (7)$$

In Equation 7, $interest_m$ is the normalized predicted outcome of the regression tree for the rule m. Finally, the last component in Equation 6 indicates the complement of the coverage of the rule among the already selected recommendations. We propose two different versions of this adapted xQuAD.

- **RT.** $p(j|m)$ is a binary function that returns 1 if the item j matches the rule, 0 otherwise.
- **DivRT.** $p(j|m)$ is the average similarity between m and each rule covered by item j. More formally:

  $$p(j|m) = \text{avg}_{m' \in M(u,j)}\, sim(m, m') \quad (8)$$

  The rationale behind this formulation is that some rules may be similar with each other thus not bringing any actual diversification if considered separately. The computation of $sim(m, m')$ takes into account the overlapping between the rules m and m' as follows:

  $$sim(m, m') = \frac{\sum_{c_i \in body(m)} overlap(m, m', c_i)}{max(|body(m)|, |body(m')|)}$$

  For instance, considering the attributes represented in Figure 1, we have for $actor, genre$ and $director$:

  $$overlap(m, m', c_i) = \begin{cases} 1, & c_i \in body(m) \wedge c_i \in body(m') \\ 0, & \text{otherwise} \end{cases}$$

  For the numerical attribute *year* we may adopt a different formulation for the function $overlap(m, m', c_i)$. Here we compute, if any, the overlap between the interval in $body(m)$ and the one in $body(m')$ normalized with respect to maximum interval's length. As an example, if $year > 1990$ is in $body(m)$ and $year < 2010$ is in $body(m')$ we may define the overlapping function as $overlap(m, m', c_i) = \frac{|1990-2010|}{max(dom(year)) - min(dom(year))}$.

  The functions introduced above have been used in the experimental setting in order to compute the function $overlap(m, m', c_i)$ (see Section 4).

RT and DivRT can be used instead of the basic xQuAD as diversification algorithms in the re-ranking phase. Alternatively, basic xQuAD and RT or DivRT can be pipelined to benefit from the strengths of them both. For instance, one could use xQuAD to select 50 diversified recommendations and then RT to select 20 recommendations from those 50, or vice versa. Hereafter, we use the syntax X-after-Y, e.g. xQuAD-after-RT, to indicate that algorithm X is executed on the results of Y.

4. EXPERIMENTS

We carried out a number of experiments to evaluate the performance of the methods presented in the Section 3 on two well known datasets: MovieLens1M and LibraryThing.

`MovieLens 1M`[1] dataset contains 1 million ratings from 6,040 users on 3,952 movies. The original dataset contains information about genres and year of release, and was enriched with further attribute information such as actors and directors extracted from DBpedia[2]. More details about this DBpedia enriched version of the dataset are available in [11]. Because not all movies have a corresponding resource in DBpedia, the final dataset contains 998,963 ratings from 6,040 users on 3,883 items. We built training and test sets by employing a 60%-40% temporal split for each user.

Moreover, we used the `LibraryThing`[3] dataset, which contains more than 2 million ratings from 7,279 users on 37,232 books. As in the dataset there are many duplicated ratings,

[1] Available at `http://grouplens.org/datasets/movielens`

[2] `http://dbpedia.org`

[3] Available at `http://www.macle.nl/tud/LT`

when a user has rated more than once the same item, we selected her last rating. The unique ratings are 749,401. Also in this case, we enriched the dataset by mapping the books with BaseKB[4], the RDF version of Freebase[5] and then extracting three attributes: *genre*, *author* and *subjects*. The subjects in Freebase represent the topic of the book, for instance Pilot experiment, Education, Culture of Italy, Martin Luther King and so on. The dump of the mapping is available online[6]. The final dataset contains 565,310 ratings from 7,278 users on 27,358 books. We built training and test sets by employing a 80%-20% hold-out split. The different ratio used for LibraryThing respect to Movielens (60%-40%) depends on its higher sparsity: holding 80% to build the user profile ensures a sufficient number of ratings to train the system.

	Movielens	**LibraryThing**
Number of users	6,040	7,278
Number of items	3,883	27,358
Number of ratings	998,963	565,310
Data sparsity	95.7%	99.7%
Avg users per item	275.57	20.66
Avg items per user	165.39	77.68

Table 1: Statistics about the two datasets

Since the number of distinct values was too large for year, actors and director attributes in MovieLens and for all the attributes in LibraryThing, we convert years in the corresponding decades and performed a K-means clustering for other attributes on the basis of DBpedia categories[7] for MovieLens and Freebase categories[8] for LibraryThing. Table 2 and 3 report the number of attribute values and clusters. The number of clusters was decided according to the calculation of the within-cluster sum of squares (*withinss* measure from the R Stats Package, version 2.15.3), that is picking the value of K corresponding to an evident break in the distribution of the *withinss* measure against the number of clusters extracted.

	Num. Values	**Num. Clusters**
Genres	19	-
Decades	10	-
Actors	14736	20
Directors	3194	20

Table 2: Statistics about MovieLens attributes

	Num. Values	**Num. Clusters**
Genres	270	30
Authors	12868	22
Subjects	2911	20

Table 3: Statistics about LibraryThing attributes

4.1 Experimental Configuration

For both datasets, we used the Bayesian Personalized Ranking Matrix Factorization algorithm (BPRMF) available in MyMediaLite[9] as baseline (using the default parameters). We performed experiments using other recommendation algorithms, but we do not report results here since they are very similar to those obtained by BPRMF.

We selected the top-200 recommendations for each user to generate the initial list P used for performing the re-ranking as shown in Algorithm 1.

Accuracy is measured in terms of Precision, Recall and nDCG, but we only report nDCG values since the trend of the other two metrics is very similar. Individual diversity is measured using ILD and α-nDCG (see Section 2) with $\alpha = 0.5$ to equally balance diversity and accuracy, while aggregate diversity is measured using both the catalog coverage – computed as the percentage of items recommended at least to one user – and the entropy – computed as in [3] to analyse the distribution of recommendations among all users. These two last metrics need to be considered together, since the coverage gives a indication about the ability of a recommender to cover the items catalog and the entropy shows the ability to equally spread out the recommendations across all the items. Hence, only an improvement of both those metrics indicates a real increasing of aggregate diversity, that in turn denotes a better personalization of the recommendations [3].

As similarity measure for computing the ILD metric (Equation 3) we used the Jaccard index. Considering that there are more attributes for each item, we computed the average of the Jaccard index value for each attribute shared between two items. α-nDCG is computed as the average of the Equation 4 for each attribute.

As presented in Section 3, we propose two novel diversification approaches: RT and DivRT. We also propose a method to combine in sequence different algorithms by means of a two phase re-ranking procedure, with the aim of benefiting from the strengths of both. Therefore we evaluated other two approaches: xQuAD-after-RT and RT-after-xQuAD, applying the second re-ranking phase on the set of 50 recommendations provided from the first phase. We have also evaluated the combination with xQuAD and DivRT, but the results are very similar using RT, so they will not be shown. To evaluate the performances, we compare the top-10 recommendation list generating from all the approaches with basic xQuAD, by varying the λ parameter from 0 to 0.95 with step fixed to 0.05 in Equation 1 (higher values of λ give more weight to accuracy, lower values to diversity).

The rules are produced using M5Rules[10] algorithm available in Weka based on the M5 algorithm proposed by Quinlan [12] and improved by Wang and Witten [22]. M5Rules generates a list of rules for regression problems using a separate-and-conquer learning strategy. Iteratively it builds a model tree using M5 and converts the best leaf into a rule. We decided to use unpruned rules in order to have more rules matchable with the items.

[4] http://basekb.com

[5] https://www.freebase.com

[6] URL removed to guarantee anonymous submission.

[7] http://purl.org/dc/terms/subject

[8] http://www.w3.org/1999/02/22-rdf-syntax-ns#type

[9] http://mymedialite.net/

[10] http://weka.sourceforge.net/doc.dev/weka/classifiers/rules/M5Rules.html

5. RESULTS DISCUSSION

Results of the experiments on MovieLens and LibraryThing are reported in Figure 2 and 3, respectively.

MovieLens. xQuAD obtains the best results in terms of ILD (Figure 2(a)) and α-nDCG (Figure 2(b)), though the xQuAD-after-RT results are very close and, with higher λ values (namely giving more importance to the accuracy factor), the differences between them are not significant. This outcome is due to the fact that the diversity metrics are attribute-based and xQuAD operates directly diversifying the attributes values, while the proposed rule-based approaches do not take into account all the attributes values. This also explains why the pure rule-based approaches (RT and DivRT) obtain the worst diversity results, while the combined algorithms (xQuAD-after-RT and RT-after-xQuAD) obtain better results. It is noteworthy that these last two configurations have no substantial difference with ILD, but, in terms of α-nDCG, xQuAD-after-RT considerably overcomes RT-after-xQuAD. This demonstrates that the pipeline of xQuAD and the rule-based approach obtains good diversity. Considering coverage (Figure 2(c)) and entropy (Figure 2(d)) to evaluate the aggregate diversity, the results show that using the rules the recommendations are much more personalized. It is interesting to note the compromise provided by xQuAD-after-RT, that obtains equidistant results between xQuAD and the rule-based algorithms, unlike RT-after-xQuAD that slightly overcomes xQuAD. With respect to the baseline, no configuration is able to give more accurate recommendations (nDCG = 0.14); all are able to increase the individual diversity (ILD = 0.34 and α-nDCG = 0.27). With nDCG and the individual diversity, the differences are always statistically significant ($p < 0.001$), except using the pure ruled-based approaches with $\lambda > 0.65$. The situation is more complex in terms of aggregate diversity, since the coverage grows very little on the baseline (coverage = 0.29) and the entropy slightly decreases (entropy = 0.78) with higher λ values. According to a comprehensive analysis on MovieLens, the pure rule-based approaches may give personalized and diversified recommendations, also with small accuracy loss. However, when individual diversity is more important than aggregate diversity, combining xQuAD with a previous rule-based re-ranking gives a good compromise between individual and aggregate diversity.

LibraryThing. At first glance, the LibraryThing results appear similar to those on MovieLens. Although they are generally consistent, there are interesting differences. Also in this case, xQuAD obtains the best diversity values, with ILD (Figure 3(a)) and α-nDCG (Figure 3(b)). However, both the combined approaches obtain really interesting results, very close to xQuAD, except for the lower λ values (namely giving more importance to the diversification factor). Unlike what happens on MovieLens, in this case RT-after-xQuAD obtains good results also in terms of α-nDCG. The pure rule-based approaches still obtain worse results. Considering coverage (Figure 3(c)) and entropy (Figure 3(d)) to evaluate the aggregate diversity, the results show that using the rules the recommendations are much more personalized than using only xQuAD. The combined approaches are able to improve the aggregate diversity with respect to xQuAD, albeit they are still distant from the pure rule-based approaches, especially in terms of coverage. With respect to the baseline, all configurations give a little more accurate recommendations, with $\lambda > 0.65$, but the differences are not statistically significant. In terms of individual diversity, all of them are able to overcome the baseline (ILD = 0.4 and α-nDCG = 0.285) except when using the pure rule-based approaches in terms of ILD. However they are able to improve α-nDCG. For the latter two metrics, the differences are always statistically significant ($p < 0.001$). In terms of aggregate diversity, xQuAD does not improve the baseline result (coverage = 0.15 and α-nDCG = 0.77), while using the rules leads to better results. According to a comprehensive analysis on LibraryThing, the pure rule-based approaches may give more personalized recommendations with a better diversity, especially using RT, with also a small accuracy loss. Similarly to the analysis on MovieLens, the results on LibraryThing suggest that diversifying with only the rules is a good choice when aggregate diversity is more important than individual diversity, conversely xQuAD remains the best choice to improve the individual diversity and combined with the rule-based diversification improves also the aggregate diversity.

The final conclusions of this analysis are that using a regression tree to infer rules representing user interests on multi-attribute values in the diversification process with xQuAD leads to more personalized recommendations but with a less diversified list and that combining attribute-based and rule-based diversifications in two phase re-ranking is a good way for taking the advantages of both. The better degree of personalization may depend on the fact that the rules are different among the users since they represents their individual interests. The lower individual diversity values with ILD and α-nDCG are due to the nature of these metrics which are based directly on the attributes values while the pure rule-based approaches do not take into account all the attributes values.

6. RELATED WORK

There is a noteworthy effort by the research community in addressing the challenge of recommendation diversity. That interest arises from the necessity of avoiding monotony in recommendations and controlling the balance between accuracy and diversity, since increasing diversity inevitably puts accuracy at risk [25]. However, a user study in the movie domain [7] demonstrates that user satisfaction is positively dependent on diversity and there may not be the intrinsic trade-off when considering user perception instead of traditional accuracy metrics.

Typically, the proposed approaches aim to replace items in an already computed recommendation list, by minimizing the similarity among all items. Some approaches exploit a re-ranking phase with a greedy selection (see Section 2), for instance [18], or with other techniques such us the Swap algorithm [23], which starts with a list of K scoring items and swaps the item which contributes the least to the diversity of the entire set with the next highest scoring item among the remaining items, by controlling the drop of the overall relevance by a pre-defined upper bound.

Other types of approaches try to directly generate diversified recommendation lists. For instance, [2] proposes a probabilistic neighborhood selection in collaborative filtering for selecting diverse neighbors, while in [16], an adaptive diversification approach is based on Latent Factor Portfolio model for capturing the user interests range and the uncertainty of the user preferences by employing the variance of

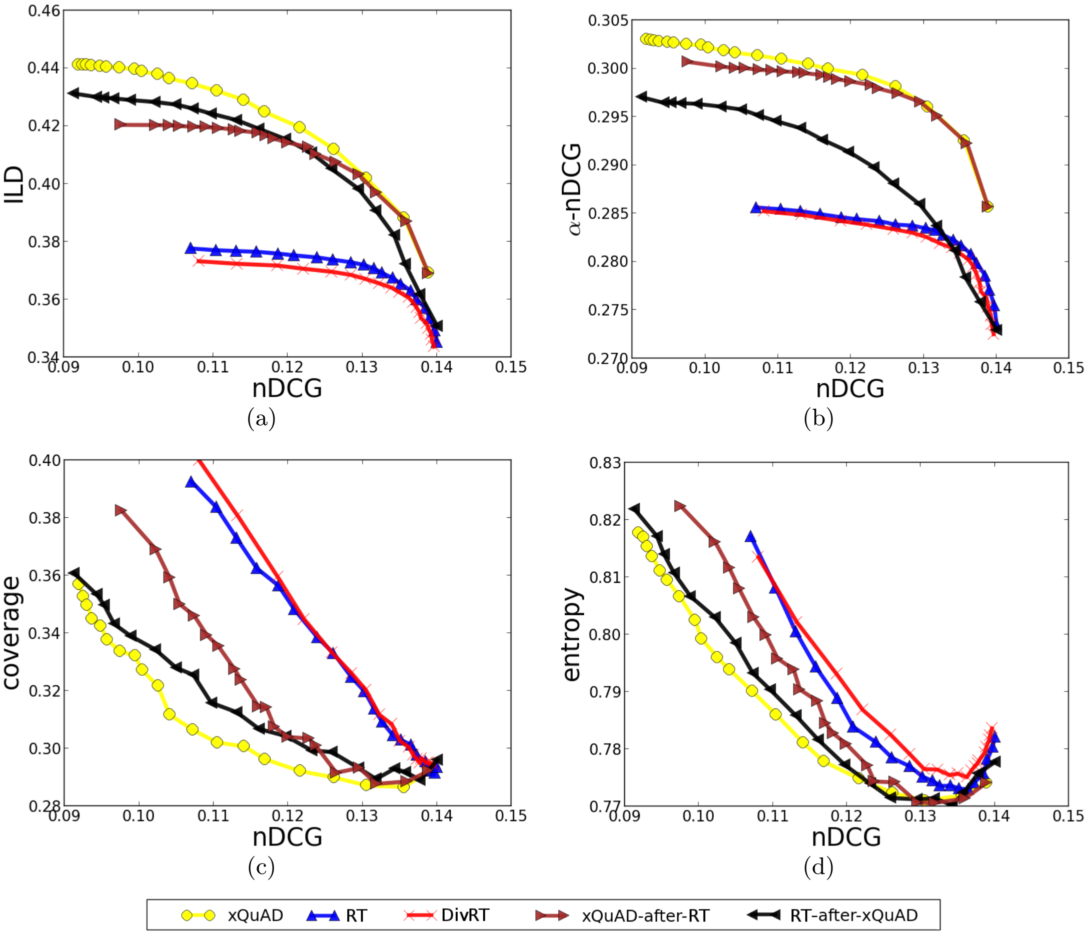


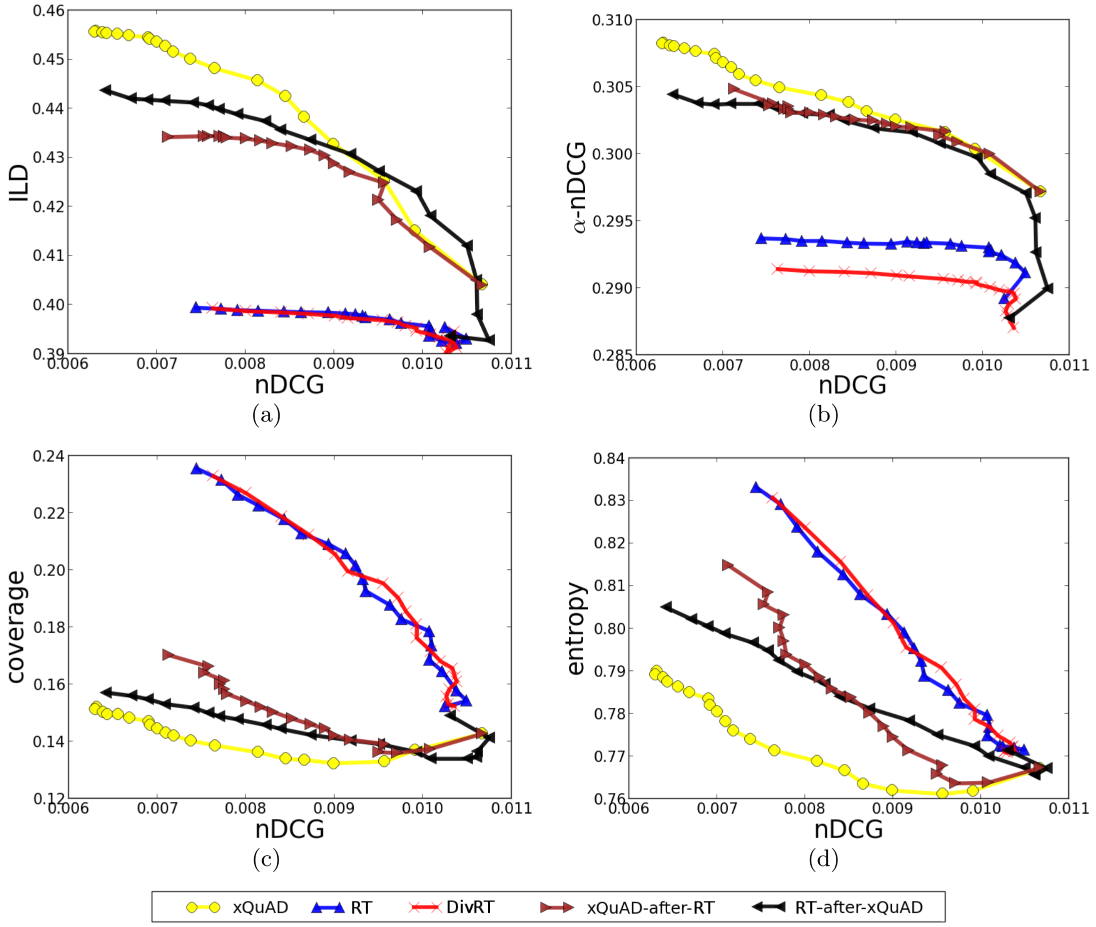
Figure 2: Accuracy-diversity curves on MovieLens at Top-10 obtained by varying the λ parameter from 0 to 0.95 (step 0.05). The statistical significance is measured based on the results from individual users, according the Wilcoxon signed-rank significance test. For nDCG and ILD 2(a), all the differences are statistically significant with ($p < 0.01$), except for those between RT and DivRT. For α-nDCG 2(b), the trend is the same, except for the differences between xQuAD and xQuAD-after-RT with $\lambda > 0.7$.

the learned user latent factors. In [13] it is proposed a hybrid method based on evolutionary search following the Strength Pareto approach for finding appropriate weights for the constituent algorithms with the final aim of improving accuracy, diversity and novelty balance. [24] considers the problem to improve diversity while maintaining adequate accuracy as a binary optimization problem and proposes an approach based on solving a trust region relaxation. The advantages of this approach is that it seeks to find the best sub-set of items over all possible sub-sets, while the greedy selections finds sub-optimal solutions.

Multi-attribute diversity has been substantially non-treated in the literature of recommender systems. A recent work [6] proposes an adaptive approach able to customize the degree of recommendation diversity of the *top-N* list taking into account the inclination to diversity of the user over different content-based item attributes. Specifically, entropy is employed as a measure of diversity degree within user preferences and used in conjunction with user profile dimension for calibrating the degree of diversification.

Furthermore, increasing attention has been paid to the intent-aware diversification, namely the process of increasing the diversity taking into account the user interests. Some approaches are based on adapted algorithms proposed for the same purpose in the Information Retrieval field, such as IA-Select [4] and xQuAD [15]. An approach for extraction of sub-profiles reflecting the user interests has been proposed in [20]. There a combination of sub-profile recommendations is generated, with the aim of maximizing the number of user tastes represented and simultaneously avoiding redundancy in the top-N recommendations. A more recent approach [19], based on a binomial greedy re-ranking algorithm, combines global item genre distribution statistics and personalized user interests to satisfy coverage and non-redundancy of genres in the final list.

The aggregate diversity, also known as sales diversity, is considered another important factor in recommendation for both business and user perspective: the user may receive less obvious and more personalized recommendations, comply with the target to help users discover new content [21] and the business may increase the sales [8]. [3] proposes the concept of aggregated diversity as the ability of a system to



(a) (b) (c) (d)

Figure 3: Accuracy-diversity curves on LibraryThing at Top-10 obtained by varying the λ parameter from 0 to 0.95 (step 0.05). The statistical significance is measured based on the results from individual users, according the Wilcoxon signed-rank significance test. For nDCG, the differences between RT and DivRT are non significant with $\lambda \in [0.2, 0.5]$. For ILD 3(a), all the differences are statistically significant with (p<0.001), except for those between RT and DivRT. For α-nDCG 3(b), all the differences are statistically significant (p<0.001).

recommend across all users as many different items as possible and proposes efficient and parametrizable re-ranking techniques for improving aggregate diversity with controlled accuracy loss. Those techniques are simply based on statistical informations such us items average ratings, average predicted rating values, and so on. [21] explores the impact on aggregate diversity and novelty inverting the recommendation task, namely ranking users for items. Specifically, two approaches have been proposed: one based on an inverted neighborhood formation and the other on a probabilistic formulation for recommending users to items. [14] proposed a k-furthest neighbors collaborative filtering algorithm to mitigate the popularity bias and increase diversity, considering also other factors in user-centric evaluation, such as novelty, serendipity, obviousness and usefulness.

7. CONCLUSIONS AND FUTURE WORK

This paper addresses the problem of intent-aware diversification in recommender systems in multi-attribute settings. The proposed approach bases on xQuAD [20], a relevant intent-aware diversification algorithm, and leverages regression trees as user modeling technique. In their rule-based equivalent representation, they are exploited to foster the diversification of recommendation results both in terms of individual diversity and in terms of aggregate one.

The experimental evaluation on two datasets in the movie and book domains demonstrates that considering the rules generated from the different attributes available in an item description provides diversified and personalized recommendations, with a small loss of accuracy. The analysis of the results suggests that a pure rule-based diversification is a good choice when the aggregate diversity is more needed than individual diversity. Conversely, basic xQuAD remains the best choice to improve the individual diversity while its combination with the rule-based diversification improves also the aggregate diversity.

For future work, we would like to evaluate the impact of our approach also on the recommendation novelty. A way to improve the novelty could be the expansion of the rules by exploiting collaborative information.

Acknowledgements. The authors acknowledge partial support of PON02_00563_3470993 VINCENTE, PON04a2_E RES NOVAE, PON02_00563_3446857 KHIRA e PON01_03113 ERMES.

8. REFERENCES

[1] P. Adamopoulos and A. Tuzhilin. On unexpectedness in recommender systems: Or how to expect the unexpected. In *in Proc of RecSys '11 Intl. Workshop on Novelty and Diversity in Recommender Systems*, 2011.

[2] P. Adamopoulos and A. Tuzhilin. On over-specialization and concentration bias of recommendations: Probabilistic neighborhood selection in collaborative filtering systems. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 153–160. ACM, 2014.

[3] G. Adomavicius and Y. Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Trans. Knowl. Data Eng.*, 24(5):896–911, 2012.

[4] P. Castells, S. Vargas, and J. Wang. Novelty and Diversity Metrics for Recommender Systems: Choice, Discovery and Relevance. In *International Workshop on Diversity in Document Retrieval (DDR 2011) at the 33rd European Conference on Information Retrieval (ECIR 2011)*, April 2011.

[5] C. L.A. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 659–666. ACM, 2008.

[6] T. Di Noia, V. C. Ostuni, J. Rosati, P. Tomeo, and E. Di Sciascio. An analysis of users' propensity toward diversity in recommendations. In *ACM RecSys '14*, RecSys '14, pages 285–288. ACM, 2014.

[7] M. D. Ekstrand, F. M. Harper, M. C. Willemsen, and J. A. Konstan. User perception of differences in recommender algorithms. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 161–168. ACM, 2014.

[8] D. Fleder and K. Hosanagar. Blockbuster culture's next rise or fall: The impact of recommender systems on sales diversity. *Management science*, 55(5):697–712, 2009.

[9] N. Hurley and M. Zhang. Novelty and diversity in top-n recommendation – analysis and evaluation. *ACM TOIT*, 10(4):14:1–14:30, 2011.

[10] S. M. McNee, J. Riedl, and J. A. Konstan. Being accurate is not enough: How accuracy metrics have hurt recommender systems. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '06, pages 1097–1101, 2006.

[11] V. C. Ostuni, T. Di Noia, E. Di Sciascio, and R. Mirizzi. Top-n recommendations from implicit feedback leveraging linked open data. In *ACM RecSys '13*, pages 85–92, 2013.

[12] R. J. Quinlan. Learning with continuous classes. In *5th Australian Joint Conference on Artificial Intelligence*, pages 343–348, Singapore, 1992. World Scientific.

[13] M. T. Ribeiro, A. Lacerda, A. Veloso, and N. Ziviani. Pareto-efficient hybridization for multi-objective recommender systems. In RecSys '12, pages 19–26. ACM, 2012.

[14] A. Said, B. Fields, B. J. Jain, and S. Albayrak. User-centric evaluation of a k-furthest neighbor collaborative filtering recommender algorithm. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 1399–1408. ACM, 2013.

[15] R. L.T. Santos, C. Macdonald, and I. Ounis. Exploiting query reformulations for web search result diversification. In WWW '10, pages 881–890. ACM, 2010.

[16] Y. Shi, X. Zhao, J. Wang, M. Larson, and A. Hanjalic. Adaptive diversification of recommendation results via latent factor portfolio. In *ACM SIGIR '12*, pages 175–184, 2012.

[17] B. Smyth and P. McClave. Similarity vs. diversity. In *Proceedings of the 4th International Conference on Case-Based Reasoning: Case-Based Reasoning Research and Development*, ICCBR '01, pages 347–361. Springer-Verlag, 2001.

[18] S. Vargas, L. Baltrunas, A. Karatzoglou, and P. Castells. Coverage, redundancy and size-awareness in genre diversity for recommender systems. In RecSys '14, pages 209–216, 2014.

[19] S. Vargas, L. Baltrunas, A. Karatzoglou, and P. Castells. Coverage, redundancy and size-awareness in genre diversity for recommender systems. In RecSys '14, pages 209–216. ACM, 2014.

[20] S. Vargas and P. Castells. Exploiting the diversity of user preferences for recommendation. In *OAIR '13*, pages 129–136, 2013.

[21] S. Vargas and P. Castells. Improving sales diversity by recommending users to items. In *Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014*, pages 145–152, 2014.

[22] Y. Wang and I. H. Witten. Induction of model trees for predicting continuous classes. In *Poster papers of the 9th European Conference on Machine Learning.* Springer, 1997.

[23] C. Yu, L. Lakshmanan, and S. Amer-Yahia. It takes variety to make a world: Diversification in recommender systems. In *EDBT '09*, pages 368–378, 2009.

[24] M. Zhang and N. Hurley. Avoiding monotony: Improving the diversity of recommendation lists. In *ACM RecSys '08*, pages 123–130, 2008.

[25] T. Zhou, Z. Kuscsik, J.G. Liu, M. Medo, J.R. Wakeling, and Y.C. Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences*, 107:4511–4515, 2010.

[26] C. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *WWW '05*, pages 22–32, 2005.