2012 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing

ARSYS - Article Recommender System

Cristian Bancu[1], Monica Dagadita[1], Mihai Dascalu[1], Ciprian Dobre[1], Stefan Trausan-Matu[1, 2], Adina Magda Florea[1]
[1] Department of Computer Science and Engineering, "Politehnica" University of Bucharest
Bucharest, Romania
[2] Research Institute for Artificial Intelligence of the Romanian Academy
Bucharest, Romania
{cristian.bancu, monica.dagadita}@cti.pub.ro, {mihai.dascalu, ciprian.dobre, stefan.trausan, adina}@cs.pub.ro

***Abstract*— In a world flooded by information, a filtering mechanism is compulsory. Recommender systems have taken a huge leap towards this goal, significantly improving the user experience in the online environment. There are two main approaches, content-based and collaborative filtering, both with advantages and drawbacks. We propose an article recommender system that integrates content based, collaborative and metadata recommendations, allowing users to select the method that best suits their needs. The first approach uses keywords in order to find similar articles, given a query or an entire document. Collaborative filtering is implemented using a P2P network in which data is distributed evenly across all peers. The last technique uses data from a semantic repository containing information about articles (e.g. title, author, domain), which can be interrogated using natural language-like queries. In addition, we present in detail the results obtained from employing the P2P network in terms of providing timely responses to the collaborative filtering technique and of ensuring reliability through data replication.**

Keywords – article recommendation system, collaborative filtering, Peer-to-Peer, semantic repository

I. INTRODUCTION

Due to the increased number of documents, more specifically articles, published on the web daily, the research process for newly available resources can become an overwhelming activity, especially for novices. This is where an article recommender system becomes useful due to its ability to provide a personalized list of suggested articles.

In general, there are two main approaches in the recommendation process: content-based filtering and collaborative filtering. The first method uses the content of an item and the previous user's preferences when computing recommendations, while the later approach tries to predict a user's affinity to a certain item, depending on other users' preferences.

The paper starts with an overall perspective regarding article recommender systems, semantic and peer-to-peer networks, all depicted in Section 2 and later on presents in detail ARSYS, an implemented article recommendation system in which all previous approaches are integrated. Section 4 addresses results and performance of the P2P network, while the last section focuses on conclusions and future work.

II. STATE OF THE ART

Recommender systems are software tools designed for providing suggestions of items that could be of interest to a certain user. The range of recommended items is very large, from what products to buy, to music that a person could listen or articles to read. This kind of systems are meant to help users that either have none or little experience to be able to evaluate certain products or simply to ease their choice, when they are dealing with a large variety of alternatives.

A. Article Recommendation systems

A very important part of regular research activities consists of reading what others have published. Due to the increasing number of documents on the web, selecting relevant materials can prove to be a cumbersome process. There are many possible methods of filtering available articles – for example, students search for guidance with their mentors who can easily evaluate the quality of an article. Furthermore, some of the factors taken into account when choosing a paper include the author(s), the journal in which it was published, the prestige of the conference at which it was presented and, of course, its actual content. This method is quite effective for a small number of articles, but the effort becomes overwhelming when there are hundreds of thousands of articles to choose from.

Even for an experienced researcher keeping pace with everything that is being published in his/her area of research can prove to become a challenge. Article recommender systems represent a feasible solution to this problem. Thus, opinions of various experts or simple readers can be used in order to select qualitative materials. The users of such a system need not to be only experts; they can be also novices in a certain domain that wish to learn, get acquainted with current approaches and need guidance.

General article recommender systems are already implemented and some of the most popular solutions will be presented in the following subsections.

1) Mendeley

Mendeley [7] has two versions (a desktop and a web-based one) that can be used either in a synchronized manner or independently. The main objectives of Mendeley address helping a user organize his/her research process and easing the discovery of new information.

Mendeley Desktop extracts metadata from user's documents by building an index for each text. This way searches are performed quicker and building bibliographic references is much easier.

Other functionalities include text highlighting, notes adding and other means of annotation. Additionally, Mendely Web allows for document upload and synchronization with the desktop component.

978-0-7695-4934-7/12 $26.00 © 2012 IEEE
DOI 10.1109/SYNASC.2012.38

Therefore Mendeley can be considered a social network specially designed for researchers that need good collaboration means and quick information access. Added to this, the system also provides content based and collaborative recommendations for articles.

2) Scienstein

Scienstein [2] was one of the first recommender systems for scientific articles. Besides the classical content-based filtering, it uses citation and author analysis, implicit and explicit ratings and two new metrics, Distance Similarity Index(DSI) and In-text Impact Factor (ItIF).

DSI computes document similarity based on distance between citations. There is a set of predefined values; for example, if two references are found in the same phrase, the DSI value is one. The same paragraph will lead to a distance similarity index of 1/2, the same section 1/4, the same chapter 1/8 and 1/16 for any other situation. ItIF is another metric, used for computing the frequency of a reference within a document. Three out of four citations will lead to a factor of 0.75 and one out of four to a 0.25 ItIF value.

3) TechLens

TechLens [8] focuses on search time improvement by learning the domains of interest of a user and by using collaborative filtering in order to produce relevant results. Each user has a profile, which is either built based on his/her publications or using data collected from queries. Moreover, a user can build one or many portfolios that can ease the organization of his/her research activity.

Another distinctive feature of this system is the ability to offer recommendations to a group of people. A collaborative profile can be created for each team and all further recommendations will be made using it as input for filtering data.

4) Citeulike

Citeulike [7] is a popular system that uses social bookmarking for sharing scientific references. Similar to folksonomies, there are three main entities: users, articles and tags. Each user has a personal collection of articles that is used when generating recommendations. There is the possibility of using the whole document collection or a single article.

B. Semantic Networks

A semantic network is a graphic notation for representing knowledge whose nodes represent concepts and whose arcs mark relations between these concepts [9]. They provide a structural representation of statements about a specific domain of interest. Additionally, the idea behind the semantic web is to annotate web content by machine-interpretable metadata and to transform currently unstructured documents into a web of data through Resource Definition Framework (RDF) [19].

The declarative graphic representation of the semantic network is used to represent knowledge and to support our recommender system to better reason with metadata regarding news articles.

In our particular case we have built a definitional network that emphasizes on subtypes or is-a relations between concept types and corresponding subtypes [9]. In this context, due to the inheritance property, we took advantage of the benefits of the generated subsumption hierarchy within our ARSYS-QL language, which will be presented in detail in section III.A.4 Recommender Engine.

C. Peer to Peer Networks

Over the Internet today, computing and communication environments are significantly more complex and chaotic than classical distributed systems, lacking any centralized organization or hierarchical control. Therefore an increasing interest in emerging Peer-to-Peer (P2P) network overlays is noticed because they provide an easily adaptable base for creating large-scale data sharing, content distribution and application-level multicast applications.

These P2P networks try to provide a series of features such as: selection of nearby peers, redundant storage, efficient search/location of data items, data permanent storage or guarantees, hierarchical naming, trust, authentication and anonymity [20]. P2P networks potentially offer an efficient routing architecture that is self-organizing, massively scalable, and robust by combining fault tolerance, load balancing with the explicit notion of locality.

In this paper we are concerned with structured P2P networks meaning that the nodes and the connection between them are arranged in a restrictive structure. In an unstructured network there is no such limitation. Hence, such structured P2P systems use the Distributed Hash Table (DHT) as a substrate, in which data object (or value) location information is placed deterministically at the peers with identifiers corresponding to the data object's unique key [21].

DHT-based systems have a property that consistently assigns uniform random identifiers to the set of peers into a large space of identifiers. Data objects are assigned unique keys chosen from the same identifier space. Keys are mapped by the overlay network protocol to a unique live peer in the overlay network. The P2P overlay networks support the scalable storage and retrieval of <key, value> pairs on the overlay network. Each peer maintains a routing table consisting of its neighboring peers' identifiers and sockets. Lookup queries or message routing are forwarded across overlay paths to peers in a progressive manner, with the ids that are closer to the key in the identifier space. Different DHT-based systems will have different organization schemes for the data objects, for their key space and corresponding routing strategies.

In theory, DHT-based systems can guarantee that any data object can be located in a small overlay hops on average *O(logN)*, where N is the number of peers in the system [22]. The underlying network path between two peers can be significantly different from the path on the DHT-based overlay network. Therefore, the lookup latency in DHT-based P2P overlay networks can be high and could affect the performance of the applications.

III. ARSYS

ARSYS is an Article Recommender SYStem that uses both content-based and collaborative techniques in order to recommend documents. Users have the possibility of searching in an 11,000 article dataset, built using the metadata from CORA [10].

Content based recommendations are made using Apache Lucene [11] and citation analysis. For each document a list of cited and citing articles can therefore be accessed. Added to this, users have the possibility of querying a semantic repository using ARSYS-QL, a natural language-like query language.

Collaborative filtering is performed using a peer to peer architecture built on top of Hazelcast [12]. There are two key factors in collaborative recommendations: user's portfolios and their rating history. Each user personalizes his/her portfolio by adding articles that are considered of special interest. When computing collective recommendations for a certain user, the first step consists of identifying the set of the most similar users. Afterwards, their portfolios are merged and thus we build the recommended article list.

A. Distributed Architecture

There are two main architectural components: the super-node and the P2P network. The super-node is the only peer that has access to the system's database. It must be the first node that enters the P2P network in order to make the initializations of the shared data. The figure below shows the main modules that are part of the super-node.

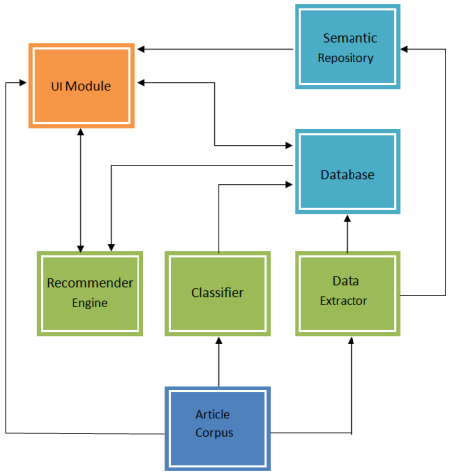

FIGURE 1. ARCHITECTURE OF A SUPER-NODE

1) Article Corpus

For our application we have used the open source CORA dataset [10] that has three main components: Citation Matching, Research Paper Classification and Information Extraction.

Cora Citation Matching groups the articles into clusters so that all articles within a cluster cite the same document. The clusters are saved in a text file with a two columns structure: the cited and the corresponding citing article. For ease of use, the file contains just articles' unique identifiers.

Cora Research Paper Classification groups most of the articles by their domain. Since a small part of the documents did not have an associated tag, we classified them as being part of a more general domain - Computer Science. Hence, a new domain was introduced in order to have all articles properly classified. Initially, there were ten main categories (without Computer Science as a more general class): Networking, Artificial Intelligence, Programming, Encryption and Compression, Hardware and Architecture, Data Structures – Algorithms and Theory, Human Computer Interaction, Databases, Information Retrieval and Operating Systems.

This information can also be found in a text file which CORA Dataset comes with and each line has the following structure: [article id] [main category] / [subcategory 1] / [subcategory 2] / ... / [subcategory n]

Cora Information Extraction component consists of multiple metadata text files (one for each article), each having a simple structure, easy to parse. For our recommendation system, we have extracted just the title, the author(s), the keywords and the abstract for each article.

Besides the incorporated open source CORA dataset, ARSYS enables the import of different article databases and the manual introduction of new articles within the article corpus.

2) Semantic Repository

In order to offer the possibility of making more specific queries, data needed to be structured in a semantic manner. In order to achieve this, we used three different ontologies that were merged manually: MAPEKUS Project domain ontology of Scientific Publications [13], Bibtex ontology [14] and DBLP ontology [15].

The figure below illustrates the way in which the hierarchy of publications is built. The presented subsumption hierarchy best suited the purposes of the research project in which ARSYS was development, but it can be easily extended or adapted to a new dataset of articles through the definition of additional / equivalent classes.

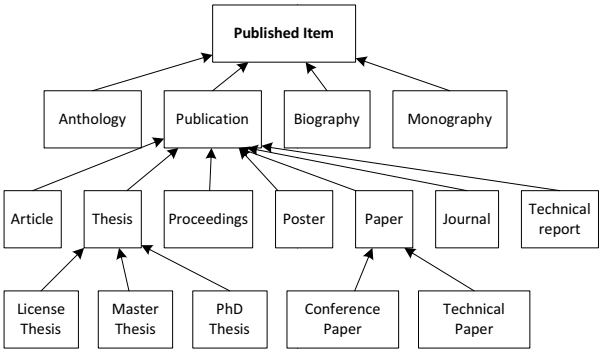

FIGURE 2. PUBLICATIONS HIERARCHY

The framework used for the creation of the semantic repository is Sesame [16]. The main reason for choosing it was the possibility of querying data using SPARQL directly - the web application offers an endpoint for such interrogations. ARSYS system uses GET HTTP requests in order to retrieve data from the semantic repository.

User queries are written in ARSYS-QL which is similar to an SQL, but more natural language oriented. Each request is processed by an interpreter which translates it to SPARQL and sends the GET request to the Sesame server. A more detailed presentation of ARSYS-QL will be made in the section describing the user interface.

3) P2P Network

The P2P network is built using [12] and links the simple peers and the super-node. Shared data is stored in a collection of maps distributed among the peers connected to the network.

Hazelcast provides both connections between peers and the distribution of the shared data [12]. In order to share data, information from the database must be saved into a map. The main source of information when computing recommendations is the user rating history. The data division between peers is made using buckets. Thus, a bucket is created for each <article,rating> pair, and the value associated to that bucket is the list of users that gave that specific rating to the article. The figure below contains a graphical representation of the mode in which information from the database was regrouped.

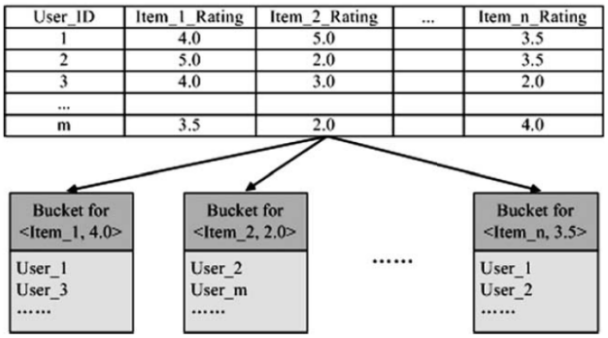

FIGURE 3. USER DATABASE DIVISION [1]

Besides ratings, we used three additional maps containing data regarding articles and user portfolios. For each article, the title and the author(s) are saved; for each user the list of articles he/she has rated and the list of documents from the portfolio are also stored locally.

Another important aspect is fault-tolerance which is achieved through the P2P overlay. The network can be configured to replicate map entries onto multiple nodes, but at least one copy is transferred to a different machine in order to ensure redundancy. Moreover, backup operations are synchronous, so when a new entry is added, it is guaranteed that the entry is replicated to at least one other node. For the read operations, the framework also guarantees that the latest value of the entry is always returned. Hence, the system provides enhanced reliability and fault-tolerance.

4) Recommender Engine

This architectural component is the main module of the application, having as purpose the computation of both content based and collaborative recommendations.

For content based recommendations, there are three options: retrieving a list of articles that cite a specific document, a list of articles that are cited within it, or the publications that are most similar from the point of view of the actual content.

The first two solutions are pretty straightforward -- they use metadata regarding citations offered by CORA [10]. For the latter, one we integrated Apache Lucene [11] and the used algorithm is composed of the following steps: firstly we determine the list of keywords from a certain article; afterwards, this list is used for making a query to Lucene, which returns a list of articles, each of them having an associated score. The next step is to give a score boost to all articles that are either in the list of citations or citing documents for the input article. Finally, documents are sorted in decreasing order of their score and are returned to the user as a list.

On the other hand, collaborative recommendations are done using the peer to peer network. The list of recommended articles is determined using a three-step algorithm: firstly, the entire rating history of user *U* is loaded from the network. Afterwards, for each <article, rating> pair we determine the associated list of users. After all pairs have been analyzed, each user is given a similarity score, depending on the number of ratings in common and similar to U. Using this score, all other users are sorted, and the top *N* are selected. For the final step of the algorithm, all portfolios of the *N* users are analyzed and the containing documents are given popularity scores, depending on the number of portfolios they were saved into. The list of recommendations will consist of the top *M* articles (sorted by popularity).

The following algorithm presents our approach for determining *N* top similar users and for selecting *M* top matching articles.

Input: *U(user id), N, M*
1. **load** ratingList hashmap
2. **for each** <article, rating> in ratingList
3. **load** userList hashmap
4. **init** userSimilarity hashmap
5. **for each** user X in userList
6. *increase similarity score*
7. **sort** userSimilarity map
8. **select top** *N* users → similarUsers map
9. **init** articlePopularity map
10. **for each** user in ratingList
11. **load** portfolio
12. **for each** article in portfolio
13. *increase popularity score*
14. **sort** articlePopularity map
15. **select top** *M* articles

The user interface was implemented using JSP [17] and Servlets [18]. There are five main pages: registration and login, content based search, collaborative recommendations, semantic search and portfolio visualization. Once the user logins or registers, he/she will be automatically redirected to the content based search page. The document corpus is then queried and the results are displayed as a list of articles - the information retrieved for each document is: title, author(s), first 500 characters, list of cited and citing articles. Added to this, there is an add button, which allows saving an article to the user's portfolio and also a like button that is equivalent to a 5-value rating (maximum value).

FIGURE 4. USER INTERFACE - CONTENT BASED SEARCH PAGE

While content based queries are done using key words, the semantic search page offers the user the possibility to specify query conditions using **ARSYS-QL**. The main advantage of this language is the fact that it is very similar to natural language.

For example, retrieval of all articles that have *Security* as domain is done with: ***SHOW article, category, title WHERE category IS Security***. The system uses a parser that analyses natural language queries, identifies certain key words like *SHOW, WHERE, IS* or *category* and transforms it to SPARQL. The translation of the previous example in SPARQL language would be:

```
1. SELECT ?a ?b ?c
2. WHERE { ?a :hasCategory ?b.
3. ?a :hasCategory ?b .
4. ?a :title ?c .
5. ?b :hasLabel ?z .
6. FILTER REGEX(?z,"security","i").
7. }
```

which is less user friendly.

Although translations based on pattern-matching rules are used to transform ARSYS-QL to SPARQL requests and the syntax of the language does not permit advanced filters similar to the SPARQL language, we consider ARSYS-QL a bridge towards a truly semantic web-portal, but without the cognitive overload of users trying to grasp the SPARQL language syntax.

Figure 5 depicts the results obtained from the previous query by relating to the articles indexed from the CORA dataset:

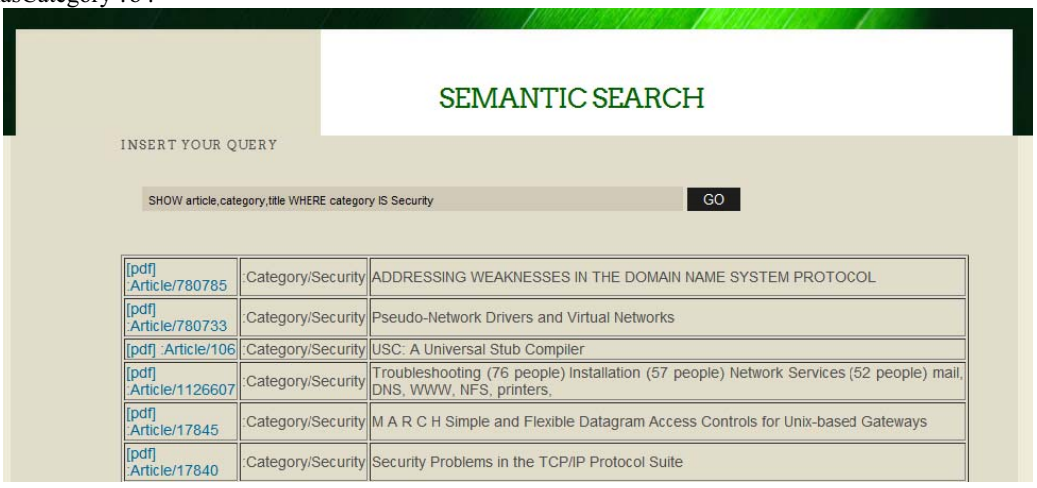

FIGURE 5. USER INTERFACE - SEMANTIC SEARCH PAGE

The Portfolio page displays all articles saved by the user. At registration time, each user is associated to a portfolio. As the user uses the add button, documents are saved to the portfolio. This data will be used when making collaborative recommendations. As previously described, user ratings and portfolio data is processed in order to determine the set of documents that are to be recommended.

RESULTS OF THE DISTRIBUTED ARCHITECTURE

This section focuses on the results obtained for the collaborative recommendation part of the system, more precisely the distribution of data among peers and the computation times for article recommendations.

In order to run the test, a set of ratings had to be generated. In order to achieve this, we created 10 users and for each user we generated 300 ratings. Two thirds of the ratings are for articles in Artificial Intelligence and one third is distributed randomly throughout the article corpus.

All tests were done using four different machines communicating through wireless. The configurations for all machines are described in the Table I.

TABLE I. HARDWARE CONFIGURATION OF TEST MACHINES

Machine	Processor	Number of cores	RAM (GB)
Super peer	Intel Core 2 Duo @ 2.26GHz	2	4
Peer 1	Intel Core i7 @ 2GHz	4	8
Peer 2	Intel Core i7 @ 2.4GHz	4	8
Peer 3	Intel Core 2 Duo @ 2GHz	2	2

Figure 6 depicts the evolution of the ratings map size at different times (1st, 3rd, 9th and 18th iteration respectively) and we can easily observe that the map is tending to become evenly distributed among all participants. During the first iteration, only the super peer is connected to the network and thus all data is found on a single node. As other peers join the network and access their list of recommendations, the four maps are being distributed among them. However, not all data is transferred from the beginning, as it takes 18 iterations until getting to an uniform distribution.

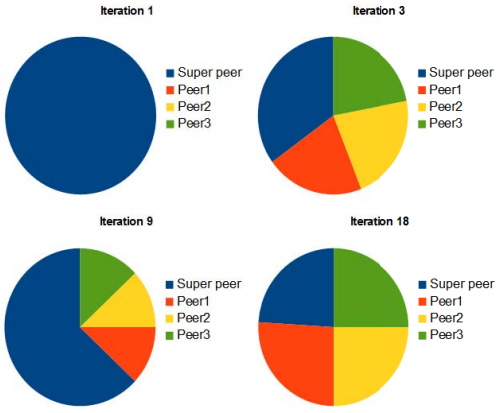

FIGURE 6. DATA DISTRIBUTION OF PEERS

Regarding the loading times of the collaborative recommendations, the super peer has the longest time to initialize due to DB readings, whereas the other peers take the information directly from the DHT overlay network.

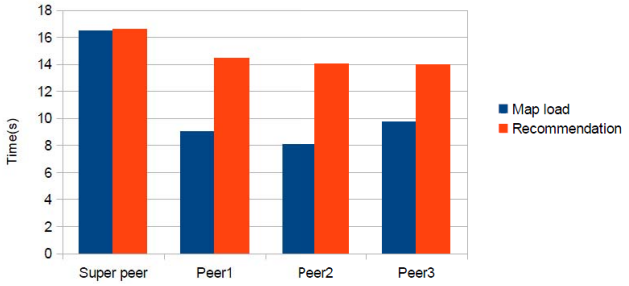

FIGURE 7. LOAD AND COMPUTATION TIMES

Considering that all the peers have just started a new instance of ARSYS, it takes approximately 17 seconds to the super peer to load the data from the database into the network and under 1 second to compute the list of collaborative recommendations, since all required information is stored locally. For the rest of the peers, the load time of data varies from 8 to 10 seconds, whereas the computation time did not exceed 3.2 seconds in the provided testing environment. The difference in the loading time between the third peer in comparison to the first two is due to its inferior hardware platform (a slower CPU with less cores and only 2 GB of RAM instead of 8. Anyway, the advantage of the distribution of data among peers is obvious, as loading times are almost reduced to half compared to the case when there is a single peer.

Below, the evolution of backups is shown. The network was configured to replicate map entries onto a single node. Therefore, when a new entry is added it is automatically replicated to another node.

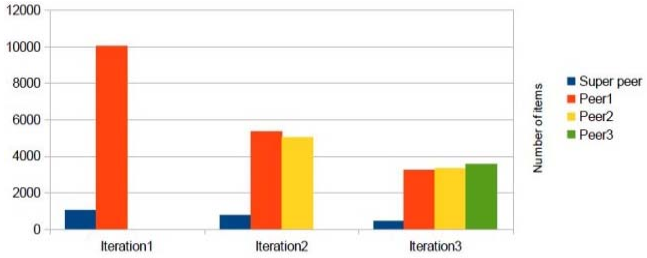

FIGURE 8. BACKUP DISTRIBUTION

The figure above shows how the article map is replicated as new peers join the network. In the first iteration, most of the initial map is saved onto the super peer node, and thus the replication is done on peer1. Afterwards, the backup distribution becomes more equitable between the peers and the super peer.

For further increasing the coverage of the results in terms of plausible usage scenarios, we plan to broaden the diversity of the applied tests by sampling different values for the number of peers, the number of users and their corresponding ratings.

IV. CONCLUSIONS AND FUTURE WORK

We have introduced a novel article recommender system that integrates multiple perspectives, from both the provided recommendation features, but also from the point of view of the integrated components, featuring semantic repositories, natural language-like query interpretation and P2P overlays for efficient computations. The presented results prove that our approach is robust and that a significant improvement in terms of performance and reliability can be observed by deploying a peer-to-peer architecture.

Moreover, in order to emphasize the importance of the implemented architecture within the ongoing research project, we plan to include all articles published within our university and to implement a semi-automatic method for extracting related articles based on co-author publishing and on all the references mentioned within the already indexed articles.

As future improvements, we are currently considering the parallelization of the peer processing since the iterations in the loops are independent and, can be easily distributed across multiple threads; moreover, we can use a parallel version of merge sort for sorting tasks.

Additionally, enabling a query to be written in natural language improves the usability and the query becomes more suggestive. The current syntax of ARSYS-QL permits selecting and filtering, without the possibility of sorting, grouping or aggregation functions. Extending the language with such features would allow an easier and more personalized interaction with the system.

ACKNOWLEDGMENT

The research presented in this paper was supported by project No.264207, ERRIC-Empowering Romanian Research on Intelligent Information Technologies/FP7-REGPOT-2010-1.

REFERENCES

[1] P. Han, B. Xie, F. Yang, R. Shen, “A scalable P2P recommender system based on distributed collaborative filtering”, 2004.

[2] B. Gipp, J. Beel, C.H. Scienstein, “A Research Paper Recommender System”, in Proceedings of the international conference on emerging trends in computing (icetic’09), pages 309-315, Virudhunagar (India), 2009. Kamaraj College of Engineering and Technology India, IEEE, 2009.

[3] C.-N. Ziegler, G. Lausen, J.A. Konstan, “On exploiting classification taxonomies in recommender systems”, 2008.

[4] K. Nageswara Rao, V.G. Talwar, “Application Domain and Functional Classification of Recommender Systems - A Survey”, DESIDOC Journal of Library Information Technology, pages 17-35, 2008.

[5] G. Adomavicius, A. Tuzhilin, “Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions”, 2005.

[6] D. Ming Chiu, “A Paper Recommendation System”, 2008, Available: http://home.ie.cuhk.edu.hk/~dmchiu/prs.pdf [May 15, 2012]

[7] D.Hull, SR. Pettifer SR, DB Kell, “Defrosting the Digital Library: Bibliographic Tools for the Next Generation Web”, in PLoS Comput Biol 4(10): e1000204. doi:10.1371/journal.pcbi.1000204, 2008.

[8] N. Kapoor, J. Chen, J.T. Butler, G.C. Fouty, J.A. Stemper, J. Riedl, J.A. Konstan, “Techlens: a researcher's desktop”, in RecSys '07 Proceedings of the 2007 ACM conference on Recommender systems, pages 183-184, 978-1-59593-730-8, 2007.

[9] J.F. Sowa, "Semantic Networks", in Stuart C Shapiro. Encyclopedia of Artificial Intelligence, 1987.

[10] Cora Dataset, Available: http://people.cs.umass.edu/~mccallum/data.html [May 15, 2012]

[11] O. Gospodnetic, E. Hatcher, M. McCandless, “Lucene in Action (2nd ed.)”, in Manning Publications. pp. 475. ISBN 1-9339-8817-7, 2009.

[12] Hazelcast, Available: http://www.hazelcast.com/ [May 15, 2012]

[13] Mapekus Ontology, Available: http://mapekus.fiit.stuba.sk/?page= ontologies [May 15, 2012]

[14] Bibtex Ontology, Available: http://zeitkunst.org/bibtex/0.2/bibtex.owl [May 15, 2012]

[15] DBLP Ontology, Available: http://swat.cse.lehigh.edu/resources/onto/ dblp.owl [May 15, 2012]

[16] Sesame Semantic Repository, Available: http://www.openrdf.org/ [May 15, 2012]

[17] Java Server Pages, Available: http://www.oracle.com/technetwork/java/ javaee/jsp/index.html [May 15, 2012]

[18] Servlets, Available: http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/ Servlets.html [May 15, 2012]

[19] W3C Semantic Web Activity, Available: http://www.w3.org/2001/sw/ [May 15,2012]

[20] R. Schollmeier, “A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications”, in proceedings of the First International Conference on Peer-to-Peer Computing, IEEE, 2002.

[21] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, I. Stoica, “Looking up data in P2P systems”, in Communications of the ACM, 2003.

[22] Oskar Sandberg, “Searching in a Small World”, Thesis for the Degree of Licentiate of Philosophy, Chapters 1 & 2, 2005.