International Journal of Database Theory and Application
Vol.7, No.4 (2014), pp. 237-248
http://dx.doi.org/10.14257/ijdta.2014.7.4.19

# RSSCube: A Content Syndication and Recommendation Architecture

Zijie Tang[1], Kun Ma[*1,2]

*[1]School of Information Science and Engineering, University of Jinan, Jinan 250022, Shandong, China*
*[2]Shandong Provincial Key Laboratory of Network Based Intelligent Computing, University of Jinan, Jinan 250022, China*
*ise_mak@ujn.edu.cn*

***Abstract***

*Content syndication is the process of pushing the information out into third-party information providers. The idea is to drive more engagement with your content by wiring it into related digital contexts. However, there are some shortages of current related products, such as search challenges on massive feeds, synchronization performance, and user experience. To address these limitations, we aim to propose an improved architecture of content syndication and recommendation. First, we design a source listener to extract feed changes from different RSS sources, and propagate the incremental changes to target schema-free document stores to improve the search performance. Second, the proposed recommendation algorithm is to tidy, filter, and sort all the feeds before pushing them to the users automatically. Third, we provide some OAuth2-authorization RESTful feed sharing APIs for the integration with the third-party systems. The experimental result shows that this architecture speeds up the search and synchronization process, and provides friendlier user experience.*

***Keywords:*** *Content syndication; content recommendation; RSS; information push; NoSQL; big data*

## 1. Introduction

As one form of Web 2.0 technology, really simple syndication (RSS), known as rich site summary, uses a family of standard web feeds to publish frequently updated information, such as blog articles and news headlines [1]. An RSS document (called feed or channel), including full or summarized text and metadata of information, enables publishers to syndicate data automatically [2]. In this way, subscribing to an RSS source removes the need for the user to manually check the web site for new content [3].

Although RSS aggregator is commanded to automatically download the new data, there are still some challenges of this approach. First, RSS feed is organized within the form of formatted XML item. Therefore, the search becomes a bottleneck issue when encountering massive data. Furthermore, current RSS products have weak ability to support cross-source search. Second, RSS is a hard-to-replicate source of information [4]. Some RSS products are designed to enable asynchronous synchronization of new and changed items amongst a variety of data sources. For example, some products implement timestamp-based incremental synchronization with some challenges [5]. Therefore, a set of algorithms followed by all endpoints to create, update, merge, and conflict resolve all items are a pressing need to handle the synchronization issue [6]. Third, although the content syndication has replaced traditional search through various categories of a website in order to find interesting articles successfully, users find it difficult to obtain the useful information along with an increasing number of subscribed RSS feeds. An intelligent recommendation algorithm is designed to filter and sort the massive feeds before pushing to users [7]. However, current RSS products lack of automatic collection and classification. To address these limitations, we propose a new architecture (called RSSCube) of content syndication and recommendation.

In the rest of the paper, we focus on the architecture of content syndication and recommendation system (RSSCube). The contributions of this paper are into several folds. First, we create a user-friendly content syndication and recommendation architecture. This can actively push valuable information that users are interested in instead of traditional pulling technology. This push technology enables that the feeds are filtered and sorted by the interests and hobbies of users. Second, we design an RSS source listener to capture the incremental feed changes of multiple RSS sources in a very short time. Third, we design schema-free documents to persist the feeds to speed up the search efficiency. Finally, we design OAuth2-authorization RESTful feed sharing APIs to be integrated with the third-party system.

The remainder of the paper is organized as follows. Section 2 discusses the background and related work, and Section 3 presents the requirements and objective of our content syndication and recommendation architecture. Section 4 introduces the architecture in detail. To reduce the complexity of this system, it is divided into source listener, feed search, feed recommendation, and OAuth2-authorization RESTful feed sharing APIs. Section 5 gives the experimental evaluation of the performance of our system. Brief conclusions are outlined in the last section.

## 2. Related Work

Currently, there are some open-source and commercial RSS products (also called reader or aggregator). Google Reader was a web-based aggregator, capable of reading RSS feeds online or offline. Due to declined usage, Google powered down this product. Feedly [8] is another alternative RSS reader, which is a better way to organize, read and share the content of RSS sources. It weaves the content from the RSS feeds of your favorite websites into a fun magazine-like experience and provides seamless integration with social networks. Reeder [9] is an RSS reader and client for Feedly, which caches articles and images from your feed. However, the cache will encounter the bottleneck issue in the case of massive feeds. Recently, some emerging RSS readers have supported the integration with social networking application researching on the recommendation algorithm [7, 10]. But the storage of this method to support millions of feeds is not clear.

However, there are some deficiencies of current RSS products. First, since it is not a good method to search the feeds in the XML file, most of current RSS products adopt the relational database as the storage engine. This will improve the query performance in the case of small data quantity. When the amount of data is large, search in a traditional relational database encountered the bottleneck. Although some RSS products enable cache to optimize the storage, the effect is not obvious in the case of big data. The query of feeds is always in the single RSS source, while it is not clear to support the cross-source search. The second limitation is lack of pre-processing before presenting to the users. Once a user subscribes to several RSS sources, it means all the untreated information is pushed to the user. This information consists of advertisements, advertorials and fake information. To improve user experience, the information should be sorted and filtered by the interest of users first.

### 2.1. Feed Synchronization

Currently, there are some patents on feed synchronization. The first is timestamp-based feed synchronization [5]. The RSS source listener intercepts RSS feed changes to extract the timestamp, and determine whether this feed is updated from the last synchronization timestamp to this current timestamp. However, this method takes much spaces to store massive time records for the subsequent synchronization. Although this method can shorten the synchronization time, it records too much timestamps in the case of frequent updates. Another is clock-based feed synchronization [11]. The publisher creates the feed by including a media content associated with therewith. The first virtual clock value is provided to the subscriber to modify the first virtual clock value when the subscriber modifies the media content associate with the web syndication item. However, this additional attribute of RSS feed does conform to RSS specification. On the contrast, we propose the synchronization method based on the unique key (composition of guid and pubDate of a feed) of the feed.

### 2.2. RSS Recommendation

Currently, researchers have focused on the recommendation algorithm on social networking [12-13]. Since RSS is content-centered rathen than social networking system, research on RSS recommendation is scarce. InterSynd [14] is simple Web client to recommend new RSS feeds to users based on what their neighbors have subscribed to. FeedMe [15] is another system that filters alerts with a combination of collaborative filtering technique and naive Bayes classifier. Moreover, Cui *et al.*, [16] firstly used probabilistic latent semantic analysis (PLSA) to discovery the topics of blog posts, then adopted Naive Bayesian algorithm to classify the blog posts. The goal is to reduce the noise caused by unwanted interruptions. In contrast, our goal is to recommend feeds to users and we focus on matching the feed contents with users' interest. In our previous work [17], we have proposed an online social mutual help architecture to recommend help feeds to the experts.

## 3. Requirement and Objective

We have the following requirements while designing a content syndication subscription architecture.

- Cross-source search: We want to ensure that the system has the ability to support search on cross RSS sources.
- High efficient search: Except supporting heterogeneous RSS sources, we want to improve the performance of feed search, but not limited to cache technology.
- Feed recommendation: We want to push the feeds most concerned to the users, and other useless information is discarded.
- Open internetwork: Since the content syndication and recommendation system we design provides the API for different clients, we want to provide an OAuth2-authorization RESTful feed sharing solution to be integrated with different heterogeneous systems.

## 4. Architecture

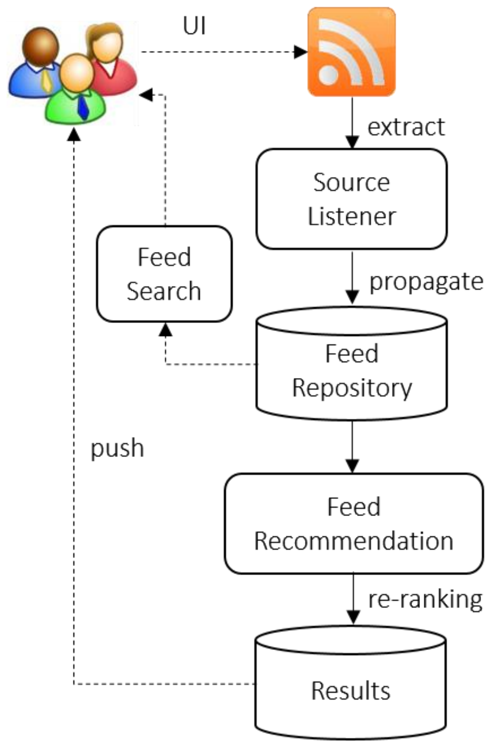


**Figure 1. Architecture of RSSCube**

In this section, we talk about the architecture of this content syndication and recommendation system, which is shown in Figure 1. The source listener, feed search, feed recommendation, and OAuth2-authorization RESTful feed sharing APIs and some key techniques are discussed in detail. First, multiple RSS sources generate a set of RSS feeds, and then source listener captures the feed updates to propagate them to the schema-free document stores. Second, we utilize TF-IDF algorithm to conclude the keywords. They are matched with the users' interest to re-rank all the recommended feeds. Finally, the results are pushed to the users.

### 4.1. Source Listener

Source listener is an important part of this system, which is responsible for propagating feed changes to the schema-free feed repository incrementally. We propose an optimized source listener to capture the increment. Initially, our source listener reads the latest information of an RSS source, and then store them in the schema-free repository for the first time. Next, this source listener only intercepts the changes to complete an incremental synchronization. In RSS 2.0 specification [1], there is an optional "guid" attribute to indicate the unique string of an RSS feed. If the feed of RSS source has this attribute, we take this as the unique key. If the feed of RSS source does not have this attribute, we take the composite source identity and timestamp as the unique key. After the RSS listener intercepts a new item, we make the incremental changes on the target schema-free repository according to this unique key. We call this source incrementality. We take polling monitoring method to intercept the frequent changes of multiple different RSS sources. We adjust the interval time on the basis of the update frequency.

## 4.2. Second and Following Pages

**Table 1. Structure of an RSS Feed**

| Field | Description |
|---|---|
| guid | A string that uniquely identifies the item. |
| title | The title of the item. |
| description | Description & The item synopsis. |
| link | The URL of the item. |
| author | Email address of the author of the item. |
| category | Includes the item in one or more categories. |
| comments | URL of a page for comments relating to the item. |
| enclosure | Describes a media object that is attached to the item. |
| pubDate | Indicates when the item was published. |
| source | The RSS channel that the item came from. |

The intercepted feeds from RSS sources are stored in the form of schema-free documents. The RSS feed has two required attributes (guid, title and description), and several optional attributes (author, pubDate, link, *et al.*,). With the rapid development of Web 2.0 sites, traditional relational databases in dealing with Web 2.0 sites encounter the bottleneck problem in the context of search. Therefore, we adopt schema-free documents to store all the updated feeds. A JSON-like document with dynamic schema is designed to store an RSS feed. The structure of this schema-free document is shown in Table 1. On one hand, separate index entry is created for each keyword (title, pubDate or category). On the other hand, full-text index is created on the large field (description) of a collection. Due to the difference of feed structure, it is difficult to implement the search on cross sources directly. Therefore, the captured feeds are stored in schema-free documents to keep a unified structure. The search issue might converted into the query on the target schema-free documents.

## 4.3. Feed Recommendation

Feed recommendation adopts a content-based recommendation technique, by mining the keywords of the contents, and matching them with the interests of subscribers'. The process by which this architecture generates a set of ranked RSS feeds is presented in detail in Figure 2.

Since there is no keyword in the feed structure, we adopt classical text-based TF*IDF algorithm [18] to conclude the keyword from the feeds automatically. The keyword extraction is conducted exploiting the TF*IDF weight of the term. It is calculated according to the formula: $TF*IDF(term)=TF(term) * \log(1+N/DF(term))$, where $TF(term)$ is the frequency of a te rm in the given feed, $N$ is the total number of feeds in the collection, $DF(term)$ is the number of feeds that contain this term.

Next, we rank RSS feeds to the subscriber. First, the keywords of each subscriber are compared the calculated keyword in the last step. We adopt Levenshtein distance algorithm [19] to measure the difference of the keywords. The objective is to find matches for short strings in many longer descriptions of the feed.

Finally, we the ranked feeds are sorted out by the Levenshtein distance to push to the users.

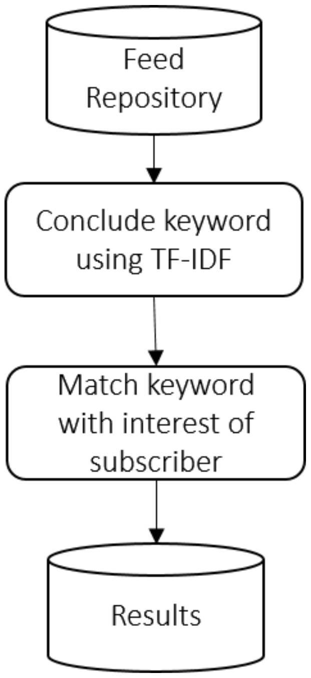

Feed Repository
Conclude keyword using TF-IDF
Match keyword with interest of subscriber
Results

**Figure 2. Process of Feed Recommendation**

### 4.4. OAuth2-authorization RESTful Feed Sharing APIs

We provide an OAuth2-authorization RESTful feed sharing APIs for the third-party system to access RSS feeds on behalf of a subscriber. Using this integration solution, users simply issue access tokens rather than the password to access their subscribed feeds for sharing. The RESTful Service is designed to obtain the subscribed feeds of a subscriber. This allows for easy integration with existing third-party system. The communication uses the form in API with the JavaScript Object Notation (JSON) format to communicate with each other. Table 2 summarizes this interface.

**Table 2. Feed Authorization API**

| Resources | URL | Parameters | Description |
|---|---|---|---|
| List of sharing feeds | *listFeeds*: /feeds/ | uid=?&token=? | find the sharing RSS feeds |

As depicted in Figure 3, the sequence starts (1) with a user requesting some service from the third-party system, and ends with the sharing feeds from our RESTful feed sharing service. The third-party system responds by (2) redirecting the end user's browser to a URL of OAuth2-authorization server. After verifying the identity of the user, the third-party system request access token from the OAuth2-authorization server. And then the third-party system can interactive with the resource server to exchange the sharing feeds.

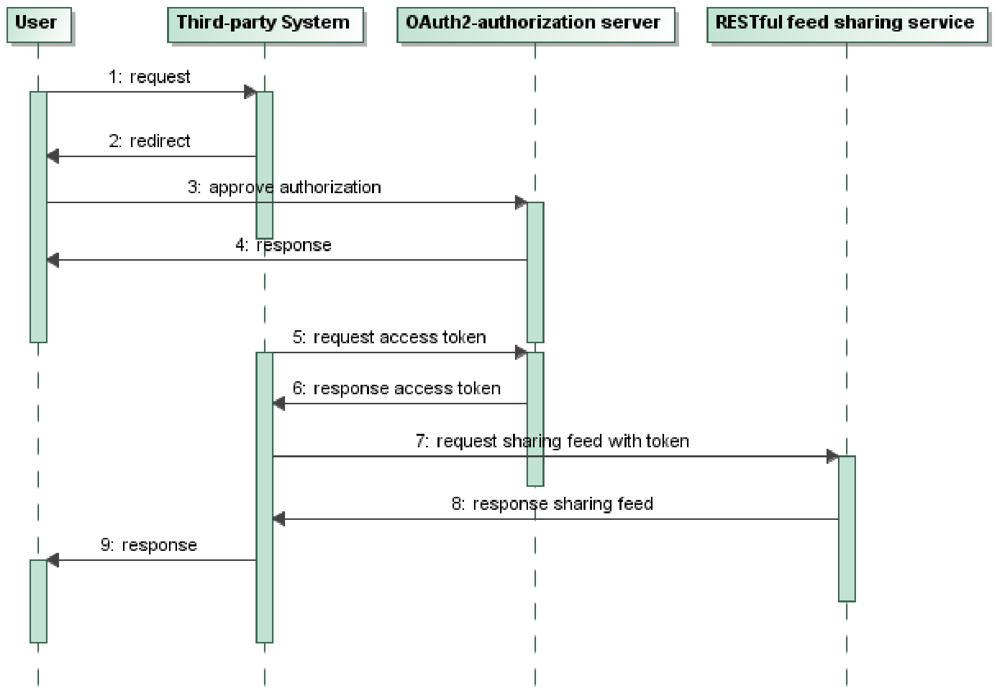
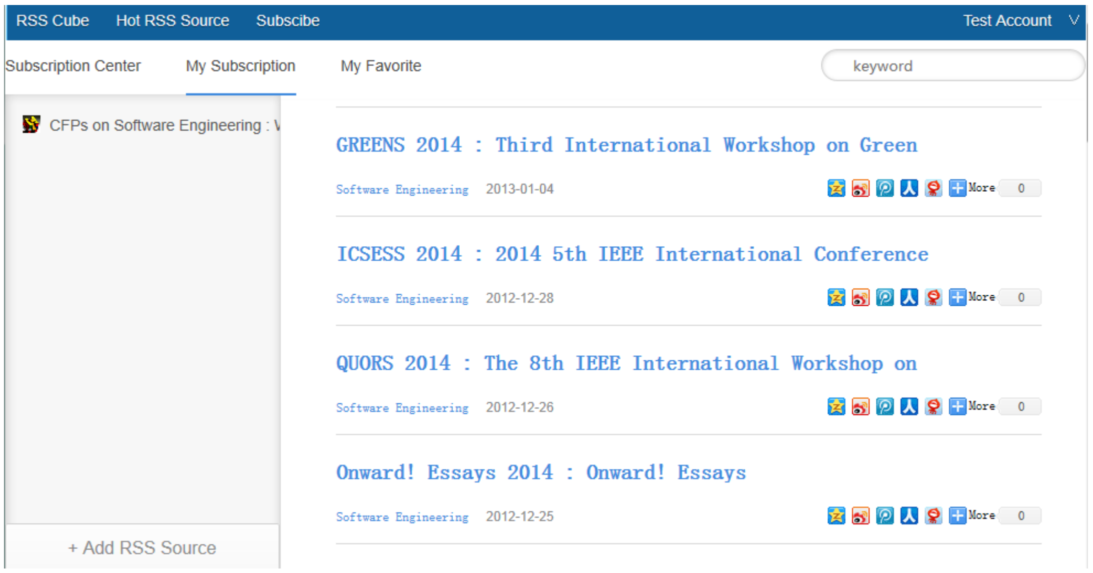


**Figure 3. Sequence Diagram of Interaction with Our OAuth2-authorization RESTful Feed Sharing Service**

## 5. Experiments



CFPs on Software Engineering : V

GREENS 2014 : Third International Workshop on Green
Software Engineering 2013-01-04

ICSESS 2014 : 2014 5th IEEE International Conference
Software Engineering 2012-12-28

QUORS 2014 : The 8th IEEE International Workshop on
Software Engineering 2012-12-26

Onward! Essays 2014 : Onward! Essays
Software Engineering 2012-12-25

**Figure 4. Screenshot of our Content Syndication and Recommendation System**

We have conducted a set of experiments to evaluate the efficiency of the proposed content syndication and recommendation architecture on our physical machine: 4 core 2.80GHz Intel Core (TM) machines with 8 GB RAM, 128G SSD and 100 Mbps Ethernet. The system was configured with a Windows Server 2012 x64. We adopt MongoDB 2.4.9 x64 as the document store. After a description of the experimental setup, we illustrate low latency of search, incremental synchronization, and user experience. We have deployed the prototype system with the link http://rsscube.duapp.com/. The screenshot of our content syndication and recommendation system is shown in Figure 4.

## 5.1. Low Latency of Search

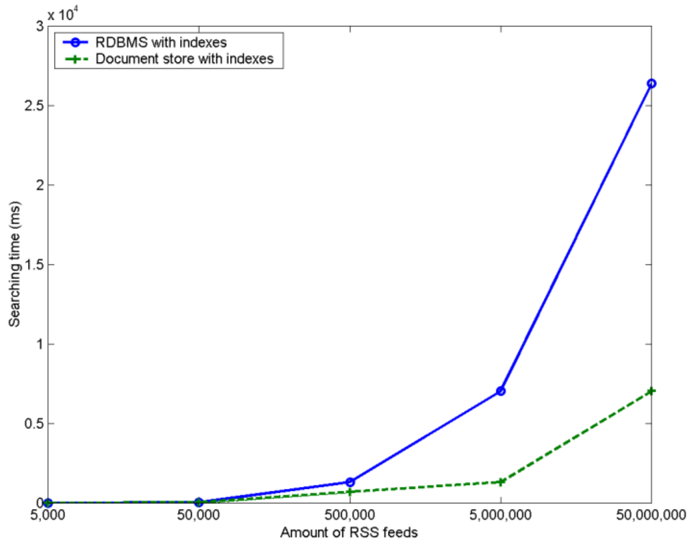


**Figure 5. Searching Time**

The first experiment is to measure the search time with different amounts of feeds. Figure 5 shows the average search time of RDBMS and document store. We issue a query "obtain the feeds in the past three months by the keywords" for many times. With less than 50,000 feeds, the search time of document store is close to RDBMS. Along with the increasing number of feeds, the average search time of document store rises more sluggishly than the RDBMS solution. That is because document stores have powerful performance on the query.

## 5.2. Incremental Synchronization

Two experiments have been made to evaluate the performance of our proposed incremental synchronization of RSS feed. First, we fix the frequency of feed updates by 1,000/s, and measure the synchronization time of two approaches in different amounts of feeds. As shown in Figure 6, the rate of synchronization time of our unique key-base method is slower with the increase of amount of feeds. Second, we fix the amount of feed by 500,000, and measure the synchronization time of two approaches in different update frequencies. As shown in Figure 7, the rate of synchronization time of our unique key-base method is slower with the increase of frequency rate.

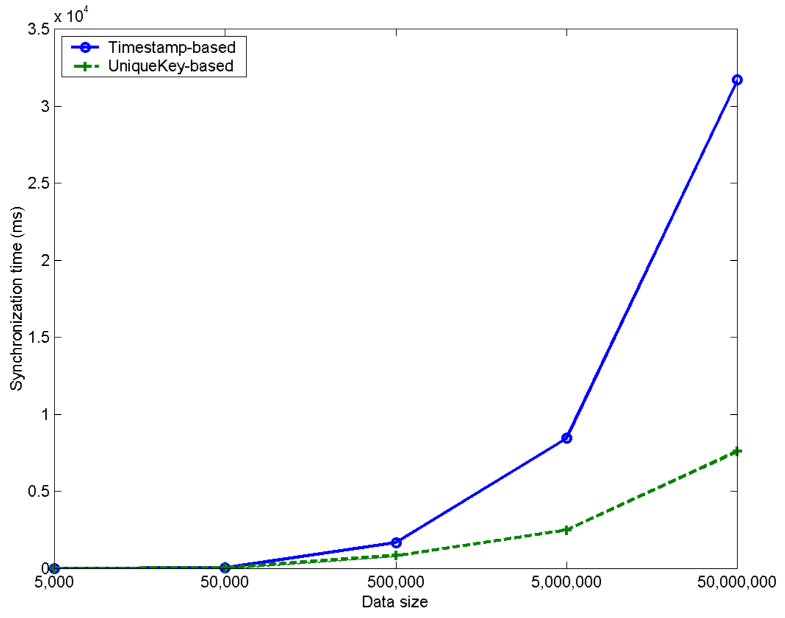


**Figure 6. Synchronization Time with the Increase of Amount of Feeds**

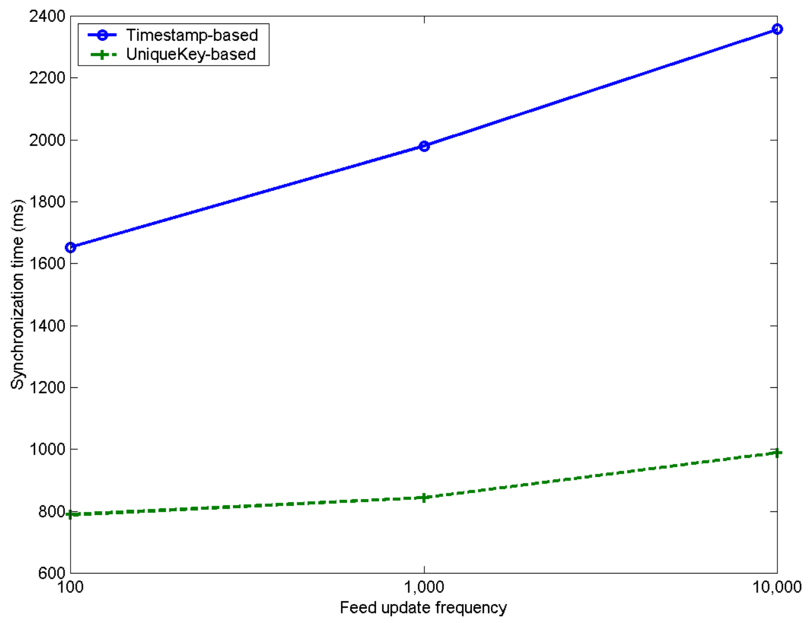


**Figure 7. Synchronization Time with the Increase of Frequency Rate**

### 5.3. User Experience

We asked them to rank the two systems: one is a famous traditional RSS reader (Due to licensing and copyright restrictions, we cannot disclose the identity of this product); the other is our system. Figure 8 shows the feedback of user satisfaction. We have received 127 valid feedback. 66.9% of users thought that our system is friendlier, 22.8% of users thought that the traditional RSS reader is better, and 10.3% of them thought our system may be better after some minor rectifications.

The approach of our solution has some advantages to user experiences. First, this active push mechanism simplifies the process of finding useful information from massive RSS feeds. Second, only the information of interest is pushed to the user. Moreover, our system re-ranks the sequence of the feeds using proposed recommendation algorithm.

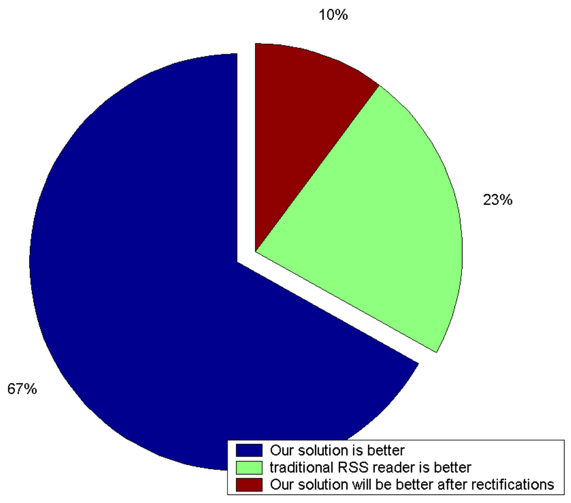


**Figure 8. User Satisfaction**

## 6. Conclusions

In an era of Web 2.0 and big data, how to access the information efficiently and effectively becomes a hot topic. Therefore, we have proposed an architecture of content syndication and recommendation, which is composed of source listener, feed search, feed recommendation, and OAuth2-authorization RESTful feed sharing APIs. Finally, experiments show that this content syndication and recommendation architecture has some features: low latency of search, incremental synchronization, and friendly user experience.

## Acknowledgements

This work was supported by the Doctoral Fund of University of Jinan (XBS1237), the Teaching Research Project of University of Jinan (J1344), and the Student Research Training of University of Jinan (SRT13310).

## References

[1] R. A. Board. (2007) RSS 2.0 speciation (version 2.0.10). [Online]. Available: http://www.rssboard.org/rss-specification
[2] S. Ovadia, "Staying informed with really simple syndication (RSS)," Behavioral & Social Sciences Librarian, vol. 31, no. 3-4, **(2010)**, pp. 179–183.

[3] M. Levy, "Web 2.0 implications on knowledge management," Journal of Knowledge Management, vol. 13, no. 1, **(2009)**, pp. 120–134.
[4] S. S. Shang, E. Y. Li, Y.-L. Wu and O. C. Hou, "Understanding web 2.0 service models: A knowledge-creating perspective", Information & Management, vol. 48, no. 4-5, **(2011)**, pp. 178–184.
[5] J. Li, J. Shen, L. Yuan and F. Zhu, "Method, apparatus and system for improving synchronization efficiency of really simple syndication service", US Patent Application Publication 13/697 795, **(2013)**.
[6] J. Ozzie, G. Moromisato, M. Augustine, P. Suthar and S. Lees. "Feedsync for atom and RSS v1.0.2", [Online]. Available: http://feedsyncsamples.codeplex.com /wikipage?title=FeedSync20for%20Atom%20and%20RSS%20%28v1.0%29%20specification, **(2010)**.
[7] C. Ji and J. Zhou, "A study on recommendation features for an rss reader", Proceedings of 2010 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, **(2010)**, pp. 193–198.
[8] E. Khodabakchia, "Feedly, a news aggregator", [Online]. Available: http://feedly.com, **(2014)**.
[9] S. Rizzi, Reeder. [Online]. Available: http://reederapp.com, **(2014)**.
[10] O. Phelan, K. McCarthy and B. Smyth, "Using twitter to recommend real-time topical news", Proceedings of the third ACM conference on Recommender systems, **(2009)**, pp. 385–388.
[11] M. Khosravy, M. Clark, O. Lee and N. Lev, "Two-way and multi-master synchronization over web syndications", US Patent Application Publication, vol. 10, no. 7, **(2010)**, pp. 653-640.
[12] S.-Y. Yoo and O.-R. Jeong, "SNS based recommendation algorithm", Proceedings of 2013 Information Science and Applications, **(2013)**, pp. 1–3.
[13] P.-H. Soh, Y.-C. Lin and M.-S. Chen, "Recommendation for online social feeds by exploiting user response behavior", Proceedings of the 22nd international conference on World Wide Web companion, **(2013)**, pp. 197–198.
[14] A. P. ORiordan and M. O. OMahony, "Intersynd: a web syndication intermediary that makes recommendations", Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services, **(2008)**, pp. 299–304.
[15] S. Sen, W. Geyer, M. Muller and M. Moore, "Feedme: a collaborative alert filtering system", Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services, **(2006)**, pp. 89–98.
[16] C. Lin, C. Wang and X. Wu, "Blog posts recommendation based on plsa and naive bayesian classification algorithm", Journal of Chemical and Pharmaceutical Research, vol. 5, no. 12, **(2013)**, pp. 851–858.
[17] K. Ma and Z. Tang, "An Online Social Mutual Help Architecture for Multi-tenant Mobile Clouds", International Journal of Intelligent Information and Database Systems, online, **(2014)**.
[18] W. Zhang, T. Yoshida and X. Tang, "A comparative study of tf*idf, lsi and multi-words for text classification", Expert Systems with Applications, vol. 38, no. 3, **(2011)**, pp. 2758C2765.
[19] V. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals", Soviet Physics Doklady, vol. 10, **(1966)**, p. 707.

## Authors

**Zijie Tang**, is an undergraduate student of Dr. Kun Ma, studying at School of Information Science and Engineering, University of Jinan. His research interests include software development of innovative applications, data intensive computing, and big data management.

**Kun Ma**, received his Ph.D degree in Computer Software and Theory from Shandong University, Jinan, Shandong, China, in 2011. He is a senior lecturer in Provincial Key Laboratory for Network based Intelligent Computing and School of Information Science and Engineering, University of Jinan, China. He has authored and coauthored over 30 research publications in peer-reviewed reputed journals and conference proceedings. He has served as the program committee member of various international conferences and reviewer for various international journals. He is the Co-Editor-in- Chief of International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM). He is the managing editor of Journal of Information Assurance and Security (JIAS) and Information Assurance and Security Letters (IASL). He is the editorial board member of International Journal of Intelligent Systems Design and Computing and Journal of Software. He is the guest editor of International Journal of Grid and Utility Computing. His research interests include model-driven engineering (MDE), data intensive computing, big data management, and multi-tenant techniques.