Journal of Computer and Communications, 2015, 3, 65-73
Published Online June 2015 in SciRes. http://www.scirp.org/journal/jcc
http://dx.doi.org/10.4236/jcc.2015.36007

CSRecommender: A Cloud Service Searching and Recommendation System

John Wheal, Yanyan Yang

School of Engineering, University of Portsmouth, Portsmouth, UK
Email: john.wheal@myport.ac.uk, linda.yang@port.ac.uk

Received 14 April 2015; accepted 25 May 2015; published 28 May 2015

Abstract

Cloud Computing and in particular cloud services have become widely used in both the technology and business industries. Despite this significant use, very little research or commercial solutions exist that focus on the discovery of cloud services. This paper introduces CSRecommender—a search engine and recommender system specifically designed for the discovery of these services. To engineer the system to scale, we also describe the implementation of a Cloud Service Identifier which enables the system to crawl the Internet without human involvement. Finally, we examine the effectiveness and usefulness of the system using real-world use cases and users.

Keywords

Cloud Computing, Search Engine, Recommendation System, Information Retrieval

1. Introduction

Cloud Computing has become a widely used term in both the technology and business industries. The term Cloud Computing has been coined as an umbrella term to describe a category of on-demand computing services [1]. The infrastructure of these services is viewed as a "cloud", from which businesses and individuals are able to access applications from anywhere in the world [2].

A number of the underlying concepts of Cloud Computing originate from the sixties, but the potential for delivering applications via the Internet has become a reality since the Internet started offering significant bandwidth in the nineties [3]. Nowadays, Cloud Computing can refer to both the applications delivered over the Internet and the systems software and hardware running in data centres that provide these services [4]. As a result, Cloud Computing is commonly described as a stack of three distinct categories: *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS) and *Software as a Service* (SaaS).

Over recent years, the boundaries between these categories have blurred. As Staten predicted in 2011, "We

How to cite this paper: Wheal, J. and Yang, Y.Y. (2015) CSRecommender: A Cloud Service Searching and Recommendation System. *Journal of Computer and Communications*, **3**, 65-73. http://dx.doi.org/10.4236/jcc.2015.36007

will continue to see further and further blurring of the lines, which will open the market to more customers but also create confusion along the way" [5]. This confusion has manifested itself in both a lack of knowledge of services and the unrecognised potential benefit Cloud Computing has to offer.

Some companies, such as Salesforce.com, have taken the concept of SaaS and cloud platforms a stage further and provide services which allow competitors to integrate their applications via an application programming interface (API) [6]. It is therefore not uncommon to select a cloud service based on its integration with others.

In 2010, Cusumano stated "SaaS is becoming the new platform for enterprise and personal computing" [6]. This statement was backed up with research by Gartner in 2010 which found that the SaaS market was exhibiting double digit growth [7]. Despite this growth, no definitive list of cloud services has been produced. This has left consumers trawling the Internet in search of the ideal service that meets their needs.

The motivation for our work is to provide an easy method for users to find and discover relevant cloud services. We have developed a cloud service identifier, search engine and recommender system called Cloud Service Recommender (CSRecommender). At the time of writing, we are not aware of any other fully automated cloud service recommender system.

The rest of this paper is organised as follows. In Section 2, we describe the problem we are trying to solve and briefly review related work. Section 3 introduces our proposed system in detail. In Section 4, we describe the system architecture and introduce the main components. In Section 5, we report on the implementation and test results of our system. We conclude in Section 6.

2. Problem Description and Related Work

For most consumers, the use of a search engine is the natural first step in the quest to find a service. Standard search engines are inefficient at finding services and prioritise results for the major providers rather than those that would best meet the user's needs. Outside the scope of search engines, traditional marketing methods again favour the larger corporations who have a large advertising budget.

The major search engines display search results in a text based layout that is difficult to analyse for suitability and, in many cases, impossible to identify a cloud service without first visiting the website.

We are trying to solve the aforementioned problem and help users to make a purchase decision regarding a cloud service. It is clear that the use of cloud services will increase over the coming years, and whilst some review services exist, these do not focus on the discoverability of new services.

In 1999, a company dedicated to connecting buyers and sellers of business software was founded called Capterra [8]. In recent years, they have primarily focused on cloud services and hold a large catalogue of such services. Although offering a recommender service, it requires human interaction at every stage and is designed for use by large businesses.

In January 2010, a vendor and platform independent business marketplace was launched called GetApp by Nubera eBusiness SL [9]. This marketplace primarily focuses on reviews of services and requires the manual addition of new services.

In December 2011, the Gwangju Institute of Science and Communication developed Cloude—a multi-criteria cloud service search engine that supported a matching algorithm based on functional, technical and cost requirements [10]. Although providing a technically advanced search engine, Cloude required the user to have expert knowledge and was therefore not suitable for users who have never used a cloud service before. It also required the manual addition of new services and provided no collaborative recommendations.

In December 2011, the University of Melbourne proposed a framework to rank cloud services based on their ability to meet the user's Quality of Service (QoS) requirements [11]. Although QoS is an important measure for companies searching for a cloud service, this is only one of many possible measures a company would consider.

The CSRecommender system addresses the problem of users being unable to find relevant cloud services that meet their needs. CSRecommender uses a hybrid recommendation system to recommend relevant cloud services to the user.

3. System Overview

The aim of the CSRecommender system is to provide a service that identifies a cloud service and recommends relevant services to the user. The system should require as little human interaction as feasibly possible and be presented to the user in a web based user interface. The CSRecommender system is separated into the following

five distinct components:

- *Crawler* to crawl the Internet for web pages that can be passed into the Cloud Service Identifier.
- *Cloud Service Identifier* to provide a website with a cloud service score used to assess the probability that it is a cloud service.
- *Indexer* to index an identified cloud service.
- *Search Engine* to search for cloud services.
- *Recommender System* to provide both collaborative and content-based recommendations.

From the users' perspective, they will visit the CSRecommender website and register for an account. Once registered, they can search for services and leave a rating from one to five. After providing the system with a number of ratings, the user will be recommended a list of services. The following real world scenarios are an indication of how the system could be used:

- An individual has never used a cloud service before and would like to discover cloud services that meet their needs. They visit the CSRecommender website and enter a search query. A ranked list of services is displayed to the user from which they are able to select a service. Once they have selected a service, a number of recommended services that integrate with that service and those that others have rated highly are displayed.
- A business already uses a selection of SaaS products but they have become increasingly frustrated with one of the services that they are using. They visit the CSRecommender website and rate a number of services that they already use. They are then recommended other services based on the similarity between their ratings and those of other users.

4. System Architecture

A high level overview of the CSRecommender system architecture is shown in the **Figure 1**.

4.1. Crawler

The purpose of the crawler is to identify undiscovered webpages that can be inputted into the Cloud Service Identifier. It can be assumed that the majority of the webpages visited are not cloud services and therefore a large number of pages will need to be crawled before a service is identified. To increase the speed of crawling, several distributed crawlers are used. Given the scale of today's Internet, it would be considered impractical to blindly follow hypertext links in pursuit of cloud services. Instead, an initial human compiled list of directories and blogs with expected links to cloud services is used before expanding to the wider Internet.

In addition to crawling the Internet, the crawler also caches a local copy of each webpage that it visits. The

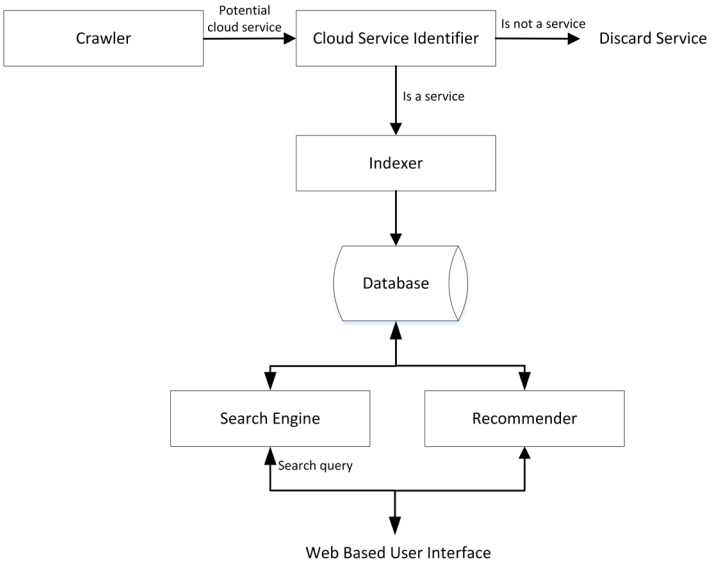

Figure 1. CSRecommender system architecture.

purpose of this is to prevent repetitive downloading of the same webpages which are required by the subsequent stages of the CSRecommender system. To prevent a build-up of data in the cache, the cache is purged once the webpage is either indexed or not identified as a cloud service.

4.2. Cloud Service Identifier

The purpose of the Cloud Service Identifier is to take a potential cloud service and assign it a score based on the certainty that it is a service. This will enable the system to crawl the Internet and grow its list of services that can be searched and recommended. Dynamically examining a cloud service would not be possible due to the majority of services requiring authentication to access. Instead, the publically accessible homepage of the service shall be used to determine whether it is a service. The principle is to compare the similarity between the homepage of a known cloud service and that of a potential one.

To achieve this, the system is first seeded with a list of URLs for known cloud services. When started, the system visits the homepage of each of these services and saves the stemmed words that appear on the page, along with each word's total frequency across all services, in a database. Any word with a frequency of one can be immediately discounted as it holds no significance.

We are most interested in words that have a high frequency but words with a low frequency should not be ignored as they may hold significance in a niche market segment. To this end, the logarithmic Equation (1) is used to compute the score of a word.

$$\text{Score} = \log\left(\text{Word Frequency}\right) \tag{1}$$

The set of words and corresponding scores can be considered the profile of an ideal cloud service. This profile can be compared to a given webpage to determine the likelihood that the webpage is a cloud service. The cosine similarity measure is used to find the similarity between the two profiles, this is shown in Equation (2) [12]. Where a webpage, *x* is compared to the ideal profile *y*, containing words with a score *r*. For example, $r_{x,s}$ is the score given to a word in the set of all words on the ideal webpage *x*. The two profiles *x* and *y* are treated as two vectors in *m-dimensional* space where $m = \left|S_{xy}\right|$.

$$sim\left(x,y\right) = \frac{\sum_{s\in S_{x,s}}\left(r_{x,s}r_{y,s}\right)}{\sqrt{\sum_{s\in S_{xy}}\left(r_{x,s}^2\right)}\sqrt{\sum_{s\in S_{xy}}\left(r_{y,s}^2\right)}} \tag{2}$$

To determine the expected similarity, the cloud services from the seed list are used to calculate their individual similarity with the ideal profile. **Figure 2** shows the number of services with a given score from a seed list of approximately 2000 services. The x-axis shows the scores while the y-axis shows the number of services given that score.

A cloud service will be identified if its score s satisfies Equation (3) where *y* is the set of seed cloud service scores.

$$s = \overline{y} \pm 4 \tag{3}$$

4.3. Indexer

A webpage that is positively identified as a cloud service is passed to the Indexer to be indexed and saved into the CSRecommender database. An overview of the Indexer architecture is shown in **Figure 3**.

A webpage is first parsed by the Indexer and the key details relating to the service extracted, including name, description and an image. The service is then saved into a database and assigned a unique ID number that is used throughout the CSRecommender system to identify the cloud service.

Lexical Analysis is performed on the parsed text from the webpage to identify each individual word. Each word is then stemmed using the Porter Stemming Algorithm [13] and the document frequency (*df*) calculated for the page.

The inverse document frequency (*idf*) is calculated on a reoccurring basis using Equation (4) [14], where *N* is the total number of services and the *idf* is calculated using the document frequency (*df*) for a term *t*.

$$idf_t = \log\left(\frac{N}{df_t}\right) \tag{4}$$

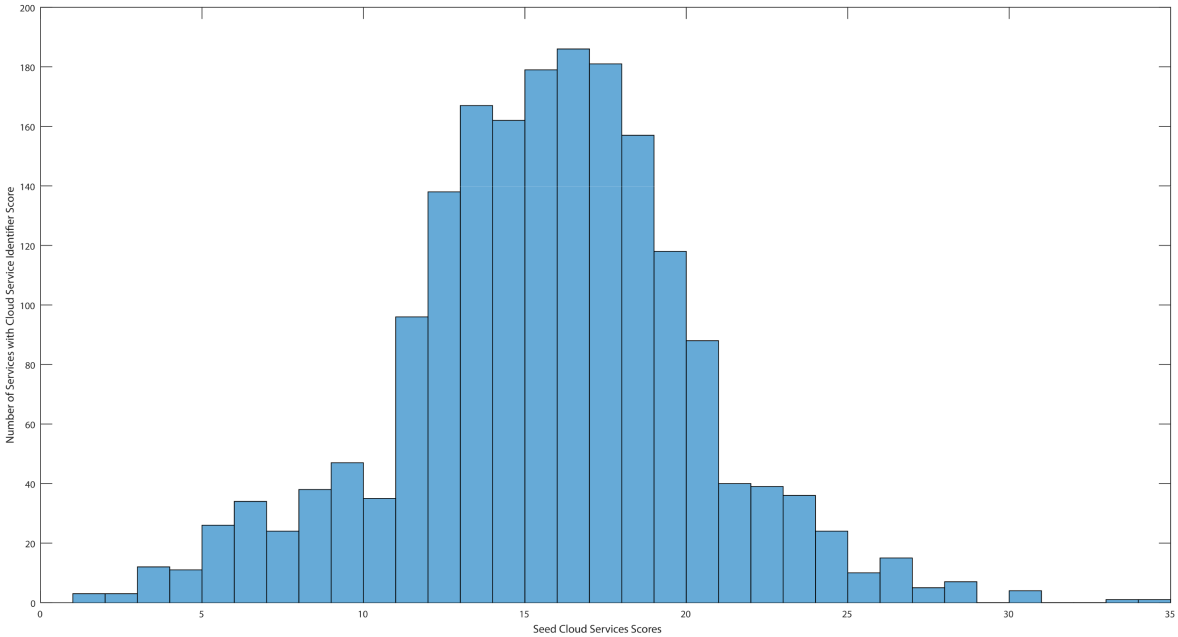

Figure 2. Cloud service identifier seed cloud service scores.

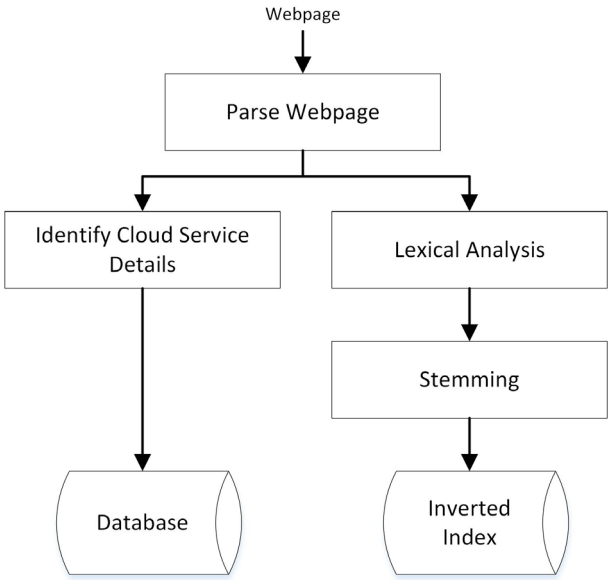

Figure 3. Indexer architecture.

The term frequency and corresponding service ID are stored as a posting in an inverted index along with the *idf* for each term, as shown in **Figure 4**.

4.4. Search Engine

The Search Engine is required to allow users to easily find cloud services based on a search term. This term can either be a feature that they require or the name of a specific service.

When a search query is received by the Search Engine, lexical analysis is performed and the resultant terms stemmed. The term frequency and inverse document frequency values from the inverted index are then used to calculate the *tf-idf* weighting for each term in each document, as shown in Equation (5).

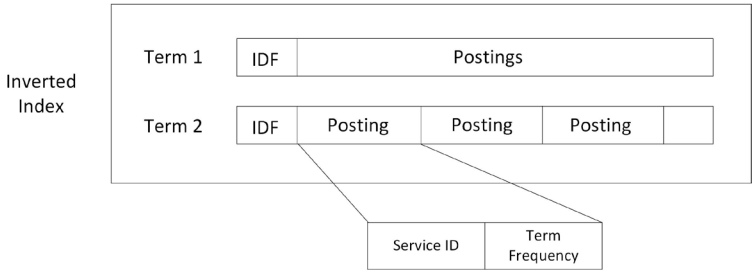

Figure 4. Indexer postings.

$$f\text{-}idf = tf_{t,d} \cdot idf_t \tag{5}$$

The Vector Space Model, shown in Equation (6), is used to rank the results [14]; where *d* is a document being compared with search query *q* and *w* the calculated *tf-idf* weight. For example, $tw_{i,j}$ is the weight given to the *ith* word in document *j*.

$$sim(d_j, q) = \frac{d_j \cdot q}{\|d_j\|\|q\|} = \frac{\sum_{i=1}^{N} w_{i,j} w_{i,q}}{\sqrt{\sum_{i=1}^{N} w_{i,j}^2} \sqrt{\sum_{i=1}^{N} w_{i,q}^2}} \tag{6}$$

Given the number of cloud services with descriptive names and the potential for users to search for a specific service, if a cloud service name includes that of the search query it is ranked at the top of the search results.

4.5. Recommender

The Recommender is the main component of the CSRecommender system and uses a hybrid approach of both content-based and collaborative recommendations. This is used to avoid the limitation of using an individual approach [12].

The content-based recommendations are calculated using two methods. The first method utilises the search query that was used by the user to navigate to the currently viewed service and recommend other services that match the search query. Although these results would not ordinarily form part of a recommender system, these results can be considered highly relevant to the user's desire to find a cloud service.

As explained in the Introduction, it is not uncommon for a user to select a cloud service based on its integration with another service. This other service would be highly relevant to the user and should therefore be recommended. Categorically determining whether a service integrates with another one would be an impossible task without human interaction. Instead, we have found that the homepage of the service usually mentions another cloud service if it integrates with it. The Recommender therefore sends the name of the cloud service currently being viewed as a search query into the Search Engine and recommends the top results, excluding itself. This is certainly not a fool-proof method of determining if a cloud service integrates with another and will have limited success for services with highly descriptive names. Although a simple method, testing has shown that in most situations it recommends highly relevant results. Although not the aim of this content-based approach, the recommendation results will still be relevant.

In addition to the two content-based approaches, the recommender also uses two further collaborative approaches to provide recommendations. Unlike content-based recommendation approaches, collaborative recommender systems attempt to predict the relevance of items for a particular user based on the items previously rated by other users [12]. To enable collaborative recommendations, the CSRecommender system allows users to rate a service based on their experience of it on a scale of one to five.

The first collaborative approach searches for users that have provided a high rating for the service that is currently being viewed by the user. Once a suitable user has been found, a selection of other services that user highly rated are displayed as recommendations.

The second collaborative approach used by the system finds users that are similar to the current user of the system. A cosine based approach [15] is used to find the similarity, where the current user *a* and the set of all other users *b* are treated as two vectors in *m-dimensional* space where $m = |S_{xy}|$. The similarity between the two vectors can be measured by calculating the cosine of the angle between them, shown in Equation (7).

$$sim(a,b) = \frac{\sum_{s \in S_{a,b}} \left(r_{a,s} r_{b,s}\right)}{\sqrt{\sum_{s \in S_{ab}} \left(r_{a,s}^2\right)} \sqrt{\sum_{s \in S_{ab}} \left(r_{b,s}^2\right)}} \quad (7)$$

Ideally, the similarity between the current user and all other users of the system should be calculated at the point when a recommendation is required. To increase the performance of the system, all user similarities are calculated in advance and only recalculated periodically.

In order to combine the content-based and collaborative recommendation results, a weighted average is calculated for each recommendation. The collaborative recommendations are most accurate when the number and agreement between the users is high [16]. In this situation, a higher weighting would therefore be given to the collaborative recommendation results.

Due to the initial user scarcity problem, the system will start by giving the content-based recommendation results a higher weighting than the collaborative results. As the number of users and ratings increase, the weighting of collaborative results will increase.

5. Implementation and Evaluation

At the time of writing, the CSRecommender system has entered its development phase. The backend system has been written in Java with a Node. js server. Approximately 2,000 cloud services have been indexed and a small number of seed users have already used the system to create an initial profile of users on which to base recommendations. The web based user interface for the system is shown in **Figure 5**.

To test the Cloud Service Identifier, fifteen cloud services were selected at random along with a further fifteen websites spanning a wide variety of genres; ranging from ecommerce to news and general information. The score was calculated for each website and checked to see whether it satisfied Equation (3). The results of the experiment are shown in **Table 1**.

The experiment has shown that the overall accuracy of the Cloud Service Identifier is 70%. Whilst promising, further work is required to increase the accuracy of the system to become commercially useful.

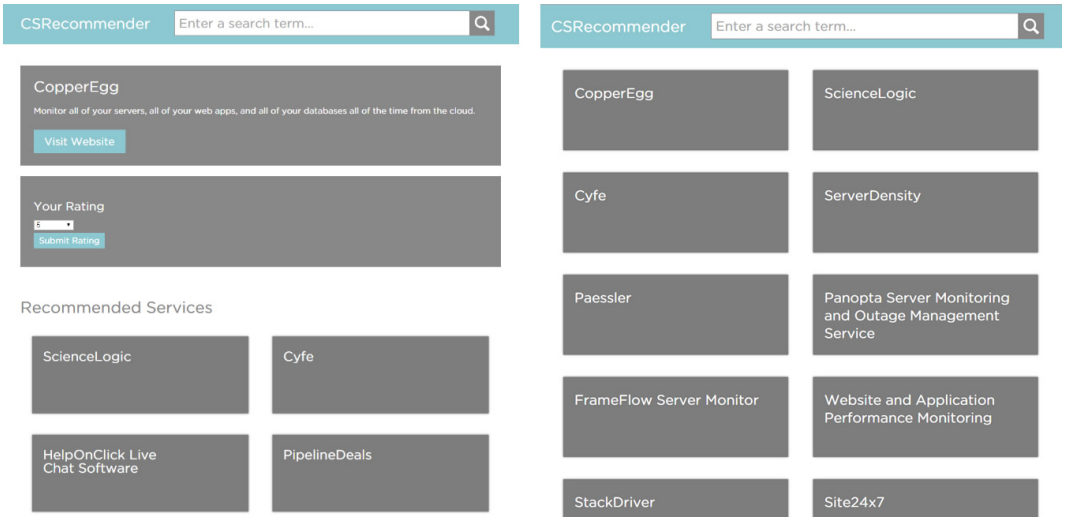

Figure 5. CSRecommender user interface.

Table 1. Cloud service identifier test results.

Website Type	Number of Websites	Identified as a Cloud Service	Accuracy (%)
Cloud Service	15	9	60
Other	15	3	80
Total	**30**	**12**	**70**

The search engine has been tested using a wide variety of search terms. Its precision has been calculated using Equation (8).

$$\text{Precision} = \frac{\#(\text{relevant items retrieved})}{\#(\text{retrieved items})}$$

Due to the scale of the system, calculating the recall for such a large data set would be impractical. Instead, the *R-Precision* is used to evaluate the search engine, where the precision is taken at the *R-th* position of the ranked results. This paper will take the value of $R = 10$. The results of three search queries are shown in **Table 2**.

The results have shown that the precision of the search engine varies depending on the nature of the search query. Queries that contain terms that are commonly associated with all cloud services is a particular area in which the search engine is weak, e.g. backup, security, storage.

To test the Recommender system, eight users with real-world use cases were asked to test the CSRecommender system. Four of the users had no experience using cloud services while the remaining four had extensive experience and were able to leave a number of ratings. This broad array of users allowed the test to cover both the use cases outlined in Section 3. The results of the experiment are shown in **Table 3**.

The experiment results show that, on average, the precision of the Recommender system is considerably higher for experienced cloud service users than those that are inexperienced. This result is as expected; experienced users will have been able to rate existing services that they use and therefore have been recommended

Table 2. Search engine test results.

Search Query	"Uptime Monitor"	Name of Popular Cloud Service	"Backup Storage"
Result 1	1.00	1.00	1.00
Result 2	1.00	1.00	1.00
Result 3	1.00	1.00	1.00
Result 4	1.00	1.00	1.00
Result 5	1.00	1.00	0.80
Result 6	1.00	1.00	0.67
Resuly 7	1.00	1.00	0.71
Result 8	0.88	1.00	0.75
Result 9	0.89	1.00	0.67
Result 10	**0.90**	**1.00**	**0.70**

Table 3. Recommender test results.

Type of User	Total Number of Recommendation Results	Number of Useful Results	Precision	Average Precision
Inexperienced Cloud Service User	4	3	0.75	**0.5**
	4	3	0.75	
	4	0	0.00	
Eexperienced Cloud Service User	4	2	0.50	**0.76**
	6	4	0.67	
	6	5	0.83	
	8	7	0.88	
	6	4	0.67	

results derived from collaborative methods. To further improve the precision of the Recommender and to prevent the system suffering from data scarcity, a clustering based approach can be used [17].

6. Conclusions

The CSRecommender system is unique in that no other fully automated cloud service indexer and recommender system currently exist. Industry predictions show that the increase in the use of cloud services by both individuals and businesses will continue to increase over the forthcoming years. The CSRecommender system will enable users to discover relevant services that meet their needs in this growing market.

The system is complex and much works remains to be done. Our immediate goals are to improve the accuracy of the Cloud Service Identifier and expand it to identify the different types of cloud service: Sass, PaaS and IaaS.

With a unique approach to identifying and recommending cloud services and the platform independent user interface, this project has the potential to become a marketable solution and certainly a system which users will find useful in the discovery of new cloud services.

References

[1] Buyya, R., Broberg, J. and Goscinski, A. (2011) Cloud Computing Principles and Paradigms. John Wiley & Sons, Inc., Hoboken. http://dx.doi.org/10.1002/9780470940105

[2] Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J. and Brandic, I. (2009) Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. *Future Generation Computer Systems*, **25**, 599-619. http://dx.doi.org/10.1016/j.future.2008.12.001

[3] Mohamed, A. (2009) A History of Cloud Computing. Computer Weekly. http://www.computerweekly.com/feature/A-history-of-cloud-computing

[4] Armbrust, M., Fox, A., Griffith, R. and Joseph, A.D. (2010) A View of Cloud Computing. *Communications of the ACM*, **53**, 50-58. http://dx.doi.org/10.1145/1721654.1721672

[5] Staten, J. (2015) Is the IaaS/PaaS Line Beginning to Blue? Forrester Research. http://www.zdnet.com/article/is-the-iaaspaas-line-beginning-to-blur/

[6] Cusumano, M. (2010) Cloud Computing and SaaS as New Computing Platforms. *Communications of the ACM*, **53**, 27-29. http://dx.doi.org/10.1145/1721654.1721667

[7] Gartner Inc. (2015) Gartner Says Worldwide SaaS Revenue within the Enterprise Application Software Market to Surpass $8.5 Billion in 2010. Gartnner. http://www.gartner.com/newsroom/id/1406613

[8] Capterra (2015) Capterra. http://www.capterra.com/

[9] Nubera eBusiness SL. (2015) GetApp. http://www.getapp.com/

[10] Kang, J. and Sim, K.M. (2010) Cloudle: A Multi-Criteria Cloud Service Search Engine. *Asia-Pacific Services Computing Conference*, Hangzhou, 6-10 December 2010.

[11] Garg, S.K., Versteeg, S. and Buyya, R. (2013) A Framework for Ranking of Cloud Computing Services. *Future Generation Computer Systems*, **29**, 1012-1023. http://dx.doi.org/10.1016/j.future.2012.06.006

[12] Adomavicius, G. and Tuzhilin, A. (2005) 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. http://dx.doi.org/10.1109/TKDE.2005.99

[13] Porter, M. (1980) An Algorithm for Suffix Stripping. *Program*, **14**, 130-137. http://dx.doi.org/10.1108/eb046814

[14] Manning, C.D., Raghaven, P. and Schutze, H. (2008) Introduction to Information Retrieval. Cambridge University Press, Cambridge. http://dx.doi.org/10.1017/CBO9780511809071

[15] Sarwar, B., Karypis, G., Konstan, J. and Riedl, J. (2001) Item-Based Collaborative Filtering Recommendation. *Proceedings of the* 10*th International Conference on World Wide Web*, Hong Kong, 1-5 May 2001, 285-295. http://dx.doi.org/10.1145/371920.372071

[16] Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D. and Sartin, M. (1999) Combining Content-Based and Collaborative Filters in an Online. Worcester Polytechnic Institute, Worcester.

[17] Pham, M.C., Cao, Y., Klamma, R. and Jarke, M. (2011) A Clustering Approach for Collaborative Filtering Recommendation Using Social Network Analysis. *Universal Computer Science*, **17**, 583-604.