Chapter 19
Analysis of Web Service Retrieval Methods

Adam Czyszczoń and Aleksander Zgrzywa

Abstract. The purpose of this chapter is to investigate different Vector Space Model approaches for Web Service Retrieval in a manner that allows the retrieval of both SOAP and RESTful Web Services. The research also includes Latent Semantic Indexing and modified term-document matrix that allows to store scores for different service components separately. All indexing models are also investigated against different weighting schemes. Additionally, this chapter introduces three test collections that were used to test all approaches in terms of effectiveness and performance. The collections can also be used for many other benchmarks on Web Service Retrieval.

19.1 Introduction

Web services are application components that are interoperable and platform independent. There exist two classes of Web Services – commonly referred to in the literature as SOAP Web Services and RESTful Web Services [1, 3, 8, 11]. Services of the first class are used mainly in the industry and lightweight RESTful Web Services are used mainly in Web applications. Developers have access to a variety of services that are published on the Internet. However, finding most suitable services from the huge collection available on the Web is still the key problem. To solve it, Web Service Retrieval techniques are used that are based on Information Retrieval models and use web crawlers to collect services that are publicly accessible on the Internet. In contrast to ineffective keyword-based service discovery methods, service retrieval can be applied to both SOAP and RESTful Web Services. The most effective and commonly used model is the Vector Space Model (VSM) that allows to model services as vectors and compare their relevancy. Another common model

Adam Czyszczoń · Aleksander Zgrzywa
Wrocław University of Technology, Institute of Informatics,
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland
e-mail: {adam.czyszczon,aleksander.zgrzywa}@pwr.edu.pl

is the Latent Semantic Indexing (LSI) that is based on VSM but allows to capture the higher-order association between services.

In this chapter we investigate current approaches to Web Service Retrieval in a model that allows the retrieval of both SOAP and RESTful Web Services. We evaluate the VSM and LSI indexing methods against different variants of *TF-IDF* (Term Frequency – Inverted Document Frequency) weighting schemes. Additionally, we present modified term-document matrix that allows to store scores for different service components separately. Such a “extended” indexing method is applied for both VSM and LSI, that gives in total four different index structures to be tested. Presented evaluation results concern effectiveness and performance. Additionally, we introduce our Web Service test collections used in evaluation and that can also be used for many other Web Service Retrieval benchmarks.

19.2 Related Work

One of the earliest research on Web Service Retrieval that used Information Retrieval methodology and utilized the VSM was presented in [10]. Authors used standard *TF-IDF* weighting scheme, inverted index file (term-document matrix) and cosine coefficient to calculate the relevance. However, authors indexed only service descriptions and depended on service repositories. The same model was followed by many other authors but they extended it with indexing more than just descriptions [12, 4], introduced grouping services into concepts [9] or proposed different indexing and ranking methods [5].

In contrast to VSM, the LSI approach allows to find hidden semantic association between terms even though they do not appear in any service. One of the earliest papers on LSI for Web Services was proposed in [7]. Introduced method used standard VSM model with *TF-IDF* inverted index file. The method, however, relied on public service repositories and concerned service descriptions only. In [13] authors presented LSI approach that also utilized standard VSM model but indexed more information than just descriptions. In further research [14] authors presented effectiveness evaluation but only for single term queries.

To sum up, any of the above research tested different weighting schemes and all of the mentioned studies were applied for SOAP Web Services only. Additionally, all the research also lacked common evaluation test collection. In many cases authors gave only the information about destination URLs of public Web Service directories but such a information quickly becomes outdated and thus not usable.

19.3 Standard Web Service Retrieval

In Vector Space Model services are represented as vectors. Every element in the document vector is calculated as weight that reflects the importance of a particular term that occurs in a particular service. The weights are computed using *TF-IDF* scheme. Weights are stored in data structure called inverted index – a term-document matrix where document vectors represent the columns and term vectors represent rows.

User query is also converted to vectors in the same manner as services. To calculate the similarity between a query and document the cosine value between this two vectors is calculated. Because vectors have different lengths they have to be normalized to the unit equal to one. Additionally, there are many different *TF-IDF* weighting variants. According to the SMART notation [6], the weighting options are denoted in a form of *ddd.qqq* syntax where first three letters represent document vector weighting and second three query vector weighting. The first letter in each triplet specifies the term frequency component of the weighting, the second the document frequency component, and the third the form of normalization used [6].

The LSI is a variant of the VSM in which the original VSM matrix is replaced by a low-rank approximation matrix using the Singular Value Decomposition. Assuming that A is a $m \times n$ matrix where $a_{ij} \in A$ represents *TF-IDF* weight of $i-th$ term in $j-th$ service, the result of the decomposition is following: $A = U\Sigma V^T$, where A is the initial VSM term-service matrix, U is the matrix which columns represent term vectors, Σ is a diagonal matrix of size $R^{terms \times services}$ containing singular values, and V is the matrix which columns represent the service vectors. By reducing matrix Σ into Σ_K as $K \times K$ matrix and projecting it on term matrix U and service matrix V^T we obtain the reduced matrix $A_K = U_K \Sigma_K V_K^T$. After reduction, the terms are represented by the row vectors of the $m \times K$ matrix $U_K \Sigma_K$, and services are represented by the column vectors the of the $K \times n$ matrix $\Sigma_K V_K^T$.

In LSI the query vectors are extracted from term matrix based on query terms. The term matrix is created during the indexing process, and thus the SMART notation *ddd.qqq* for LSI is equal to *ddd.ddd*.

19.4 Extended Web Service Retrieval

The definition of Web Service structure was elaborated in our previous study [2]. We consider Web Service to be composed of quadruple of elements α where the first three represent service *parameters* which correspond to service *name*, *description* and *version*, and the fourth element represents service *components* which are composed of six-tuple of following elements: *name*, *input value*, *output value*, *description*. To simplify we join all components as one bag-of-words. The biggest advantage of presented structure is that it conforms to both SOAP and RESTful Web Services.

All current retrieval approaches treat all service elements as single bag-of-words. The extended retrieval approach means indexing each service element α separately. Based on our research carried out in [2] Web Services are indexed in the following manner: the extended index A is a $m \times n$ matrix where $a_{ij} \in A$ is a four-tuple $< \alpha_{ij1}$, α_{ij2}, $\alpha_{ij3}, \alpha_{ij4} >$ where α_{ijx} represents weights of $i-th$ term for $j-th$ service's $x-th$ element. Weights are calculated using the modified *TF-IDF* scheme presented in Equation 19.1.

$$TFIDF_{extended} = (1 + log(\mathrm{tf}(t,\alpha))) \cdot log \frac{N_\alpha}{|\{\alpha \in WS : t \in \alpha\}|} \tag{19.1}$$

where $\mathrm{tf}(t,\alpha)$ is term frequency of term t in element α, N_α is the total number of α elements, and $|\{\alpha \in WS : t \in \alpha\}|$ is the number of service elements where the term t appears. Because service element counters are stored separately, there is substantial difference if weights between standard VSM indexing model and the extended one. Afterwards the extended index is reduced to the size of the basic VSM index structure. This is achieved using the Mean Weight Vector (MWV) method presented in [2] that is computed as average weight of all service elements. The final index structure is following: A' is a $m \times n$ matrix where $a_{ij} \in A'$ represent the MWV weight $\sigma_{ij} = \frac{1}{4}\sum_{x=1}^{4} \alpha_{ijx}$.

19.5 Test Collections

In this section we present test collections that will be used in our evaluation[1]: *SOAP_WS_12*, *SOAP_WS_14*, *SOAP_WS_12-14*. Collections are prepared using web crawlers. First collection contains 267 Web Services collected from `xmethods.net` – a directory of publicly available Web Services, used by many researchers in various benchmarks. Second collection contains 662 services collected from: `xmethods.net`, `service-repository.com`, `webservicex.net`, `venus.eas.asu.edu`, `visualwebservice.com` and `programmableweb.com` – popular repository of public Web Services. Third collection represents merged collections one and two, including elimination of duplicate services. In Table 19.1 we present summary data of the above collections.

Table 19.1 Test collections summary

	SOAP_WS_12	SOAP_WS_14	SOAP_WS_12-14
Services	267	662	889
Parameters	432	886	1254
Components	5140	18353	22660
Total elements	5572	19239	23914

At present, introduced in this chapter test collections do not include any RESTful Web Services because our methods of their identification on the Web are still being improved and currently developed collection is too small. This, however, does not influence experimental results presented in this chapter.

19.6 Evaluation

Based on the approach presented in this chapter we implemented indexing system that allowed us to conduct evaluation experiments. The evaluation concerns performance and effectiveness analysis of basic and extended LSI and VSM indexes for *ltc.ltc*, *ltc.lnc*, *lnc.lnc*, *lnc.ltc*, *mtc.ltc*, and *mtc.lnc TF-IDF* schemes, according

[1] All test collections are available to download at:
`http://www.ii.pwr.edu.pl/~czyszczon/WebServiceRetrieval`

to the SMART notation. The *lnc* variant for term weighting is a desired one, since it allows to skip some computation during the indexing process and reduce indexing time. In order to evaluate the effectiveness, we used the *WS_SOAP_12* test collection and the following classical information retrieval metrics: *Precision*, *Recall*, *F-measure* ($\beta = 1$) and *Mean Average Precision (MAP)*. In order to evaluate performance we checked the retrieval time, indexing time and index size for *WS_SOAP_12*, *WS_SOAP_14*, *WS_SOAP_12-14* test collections of every index structure. The experiment was carried out for following queries: *"temperature conversion"*, *"email validation"*, *"weather forecast"*, *"weather forecast service"*, *"currency exchange"*, *"demographics"*, *"new york"* and *"send sms"*. The MAP is computed as the average precision for all mentioned queries, whereas the average precision value is calculated for each of the *k* top relevant services in search results list.

19.6.1 Effectiveness Analysis

Based on our preliminary studies on LSI for Web Service Retrieval, we established that dimension reduction $K = 41$ gives the best effectiveness at possibly low indexing time and index size. Therefore, evaluated in this chapter LSI index is reduced to 41 dimensions. Selecting the best indexing model is evaluated using MAP at top 100 positions.

In Figure 19.1 we present the evaluation of potentially most effective *ltc.ltc* scheme against the *ltc.lnc*. Since for LSI the SMART natation is *ddd.ddd*, presented figure does not contain the LSI index for *ltc.lnc* (the LSI of *ltc.ltc* is the same as LSI of *ltc.lnc*). Both basic and extended LSI indexes were much less effective than the VSM, where for the VSM variants the worst one was the *VSM Extended ltc.lnc*. The basic version of *VSM ltc.lnc* returned less relevant services than *ltc.ltc* variants within the top 2 results, however, later it returned slightly more relevant results. Both *VSM ltc.ltc* variants performed very close to each other, with tiny difference at 19-th position on the favor of the basic structure. In general, the extended LSI was much more effective than the basic version, and the *VSM Extended ltc.lnc* was the worst method among the VSM.

In Figure 19.2 we present evaluation of *lnc.lnc* and *lns.ltc* schemes. Again LSI methods have much lower precision than VSM methods. All LSI methods performed almost at the same level. Both *VSM lnc.ltc* had the same results and had higher effectiveness than *VSM lnc.lnc*. In the case of *VSM lnc.ltc* indexes, the extended version was slightly more effective. The worst methods among the VSM were the *VSM Basic lnc.lnc* and *VSM Extended lnc.lnc*.

In Figure 19.2 we present evaluation of *mtc.ltc* and *mtc.lnc* schemes. One more time LSI methods have much lower precision than VSM methods. All VSM indexes performed very close to each other, with small advantage of the extended indexes. However, *mtc.lnc* approaches had higher overall MAP, in the beginning at top 2 positions, they resulted in lower recall. The worst method among VSM was *VSM Basic mtc.ltc*, then *VSM Extended mtc.ltc*.

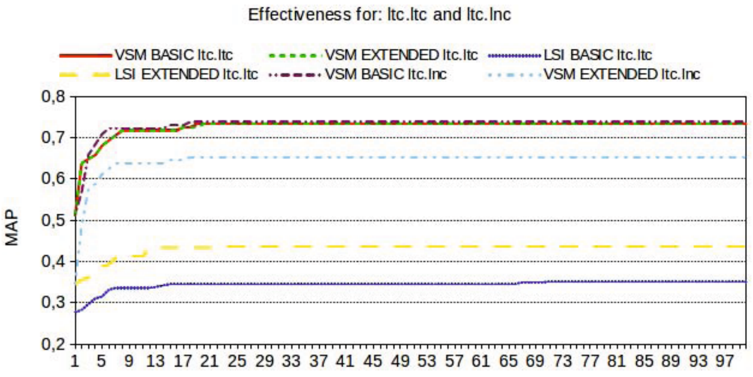

Fig. 19.1 MAP at top 100 positions for *ltc.ltc* and *ltc.lnc* schemes

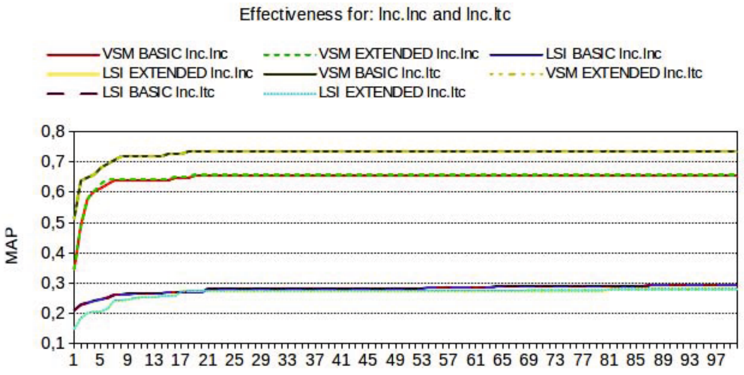

Fig. 19.2 MAP at top 100 positions for *lnc.lnc* and *lnc.ltc* schemes

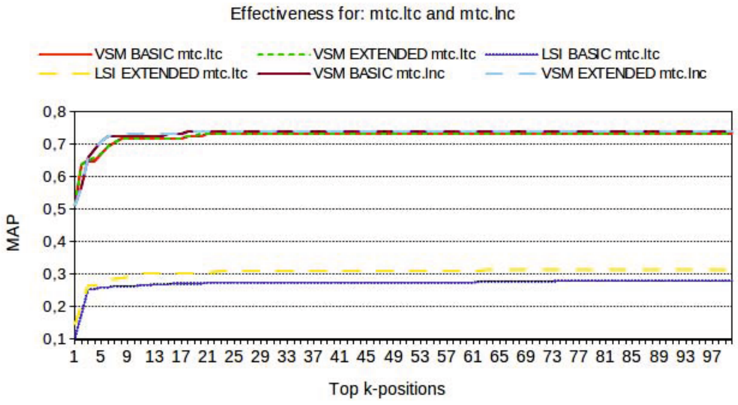

Fig. 19.3 MAP at top 100 positions for *mtc.ltc* and *mtc.lnc* schemes

To sum up, the best results for LSI index were obtained for *ltc.ltc* scheme where the extended index achieved much higher effectiveness than the basic one. In the case of VSM index, the best results were obtained by *VSM Basic lnc.ltc* and *VSM Extended lnc.ltc* equally, and *VSM Extended mtc.lnc*. In Figure 19.4 we illustrate 7 methods that obtained best results in terms of MAP. Those methods, together with both *LSI ltc.ltc* will be considered in further analysis.

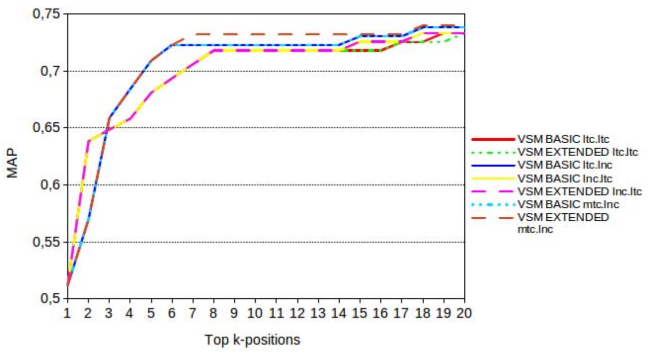

Fig. 19.4 Best seven VSM methods at top 20 positions

19.6.2 Performance Analysis

In order to investigate the performance of selected above most effective methods, we tested them against indexing time, index size, and retrieval time, for three test collections: *WS_SOAP_12*, *WS_SOAP_14*, *WS_SOAP_12-14*. In Figure 19.5 we illustrate the indexing time and index size. Both LSI methods are characterized by the longest indexing time (except for a *WS_SOAP_12* dataset) and very small index size. The indexing time of extended approach was much higher than in basic approach. However, the index size of extended approach was a little smaller than in the basic one. In the case of the VSM methods, the indexing time of extended approach was also much bigger than in the basic one. The index size was almost the same. The *lnc* and *mtc* performed much better than *ltc*. The shortest indexing time was obtained by the basic methods, where the *VSM BASIC mtc.lnc* was the best one and the *VSM Basic lnc.ltc* was the second one. Extended versions of these indexes performed 300%-425% slower, however indexing using *VSM Extended lnc.ltc* was around 10%-15% faster than *VSM Extended mtc.lnc*.

In Figure 19.6 we present the average indexing time. Both LSI methods had the fastest retrieval time for every test collection, however the extended approach was around 4%-7% faster than the basic approach. In the case of VSM methods, the retrieval time for such a small dataset as *WS_SOAP_12* was almost the same. The difference can be noticed for bigger datasets. The 3 best results were archieved by the *VSM Extended mtc.lnc*, *VSM Basic ltc.lnc*, *VSM Extended ltc.ltc* methods. The worst results were obtained by *VSM Basic mtc.lnc* and *VSM Extended lnc.ltc*. On the other hand, the differences are very small.

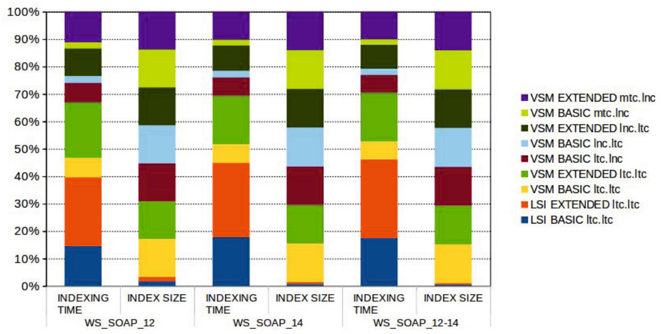

Fig. 19.5 Average indexing time and average index size of nine best indexing methods

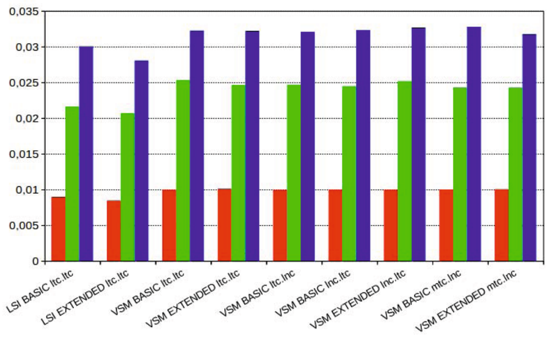

Fig. 19.6 Average retrieval time of nine best indexing methods

19.6.3 Effectiveness Comparison of Best Retrieval Methods

Based on the above effectiveness analysis and performance analysis, from selected best nine methods, we excluded from further research following approaches: *VSM Extended ltc.ltc* – it had lowest effectiveness among VSM methods and the longest indexing time, whereas, its retrieval time was comparable to other approaches; *VSM Basic ltc.ltc* – second smallest effectiveness, indexing and retrieval time worse than in other methods with higher effectiveness; *VSM Basic ltc.lnc* – effectiveness the same as in the *VSM Basic mtc.lnc* but significantly higher indexing time; *VSM Extended lnc.ltc* – effectiveness the same as in *VSM Basic lnc.ltc* but much higher indexing time. In Figure 19.7 we present five remaining methods plotted on a precision-recall curve.

In the beginning the *VSM Basic lnc.ltc* outperforms other indexing methods, but at some level (after 2nd position – see Figure 19.4) its effectiveness drops down. In other words, this method returns most relevant results on the very top of the search results list, but after second position the search results are worse than in other VSM approaches. Moreover, at some point (after 6th position – see Figure 19.4) the *VSM Extended mtc.lnc* outperforms the basic *mtc.lnc* approach. In the case of LSI, the extended approach gives more relevant results at each point.

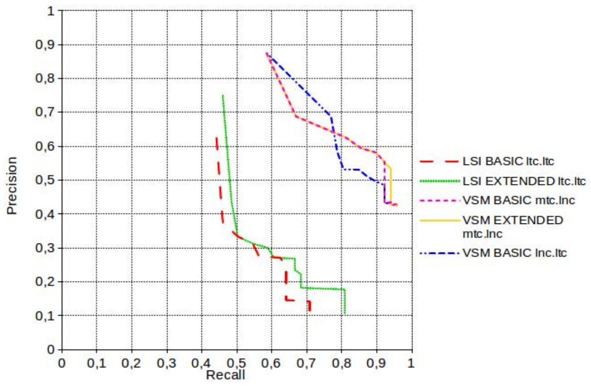

Fig. 19.7 Precision-recall curve of four best indexing methods

19.7 Conclusions and Future Work

The goal of presented research was to investigate different VSM approaches for Web Service Retrieval, including alternative approach that used modified index structure. The modified approach included extended matrix that allowed to count scores for different service components separately. Moreover, we presented three datasets that allowed us to test all approaches in terms of effectiveness and performance.

The experimental results indicate that best retrieval results can be achieved by basic *lnc.ltc* index or basic and extended *mtc.lnc* indexes, depending if we want to focus on returning most relevant results at top two positions or at top six positions. In terms of indexing time all methods, except the extended one, obtained best results. On the other hand, the extended approach turned out to be more effective than the basic one after 6th position, which is a very good result. The best *TF-IDF* scheme for the LSI methods was the *ltc.ltc*. However, the extended index proved to be much more effective than the basic LSI. Despite the fact that LSI approach return less relevant results, it captures the latent semantics between services and thus allows to return results where nothing would be found using VSM method. Therefore, it can be used as an alternative approach, when there are no results after using VSM index.

In further research we plan to check effectiveness for more queries and on all three test collections. We also plan to investigate more indexing methods and *TF-IDF* variants. For example, some researchers on Web Service Retrieval suggested that skipping length-normalization for LSI may bring better results. Our goal is to find or propose indexing method that gives highest effectiveness explicitly and that has possibly lowest indexing time and index size.

References

1. Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., Orchard, D.: Web services architecture. W3C Working Group Note 11 February 2004, World Wide Web Consortium (2004), `www.w3.org/TR/ws-arch` (accessed April 2014)

2. Czyszczoń, A., Zgrzywa, A.: The concept of parametric index for ranked web service retrieval. In: Zgrzywa, A., Choroś, K., Siemiński, A. (eds.) Multimedia and Internet Systems: Theory and Practice. AISC, vol. 183, pp. 229–238. Springer, Heidelberg (2013)
3. Erl, T., Bennett, S., Schneider, R., Gee, C., Laird, R., Carlyle, B., Manes, A.: Soa Governance: Governing Shared Services On-Premise and in the Cloud. The Prentice Hall Service-oriented Computing Series from Thomas Erl. Prentice Hall (2011)
4. Hao, Y., Cao, J., Zhang, Y.: Efficient ir-style search over web services. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 305–318. Springer, Heidelberg (2009)
5. Hao, Y., Zhang, Y., Cao, J.: Web services discovery and rank: An information retrieval approach. In: Future Generation Computer Systems, vol. 26, pp. 1053–1062 (2010)
6. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, New York (2008)
7. Paliwal, A.V., Adam, N.R., Bornhövd, C.: Web service discovery: Adding semantics through service request expansion and latent semantic indexing. In: IEEE SCC, pp. 106–113. IEEE Computer Society (2007)
8. Pautasso, C., Zimmermann, O., Leymann, F.: Restful web services vs. big web services: Making the right architectural decision. In: 17th International WWW Conference, Beijing (2008)
9. Peng, D.: Automatic conceptual indexing of web services and its application to service retrieval. In: Jin, H., Rana, O.F., Pan, Y., Prasanna, V.K. (eds.) ICA3PP 2007. LNCS, vol. 4494, pp. 290–301. Springer, Heidelberg (2007)
10. Platzer, C., Dustdar, S.: A vector space search engine for web services. In: Proceedings of the 3rd European IEEE Conference on Web Services (ECOWS 2005), pp. 14–16. IEEE Computer Society Press (2005)
11. Richardson, L., Ruby, S.: RESTful Web Services: Web Services for the Real World. O'Reilly Media, Inc., Sebastopol (2007)
12. Wu, C., Chang, E.: Searching services "on the web": A public web services discovery approach. In: Eighth International Conference on Signal Image Technology and Internet Based Systems, pp. 321–328 (2007)
13. Wu, C., Chang, E., Aitken, A.: An empirical approach for semantic web services discovery. In: Australian Software Engineering Conference, pp. 412–421. IEEE Computer Society (2008)
14. Wu, C., Potdar, V., Chang, E.: Latent semantic analysis – the dynamics of semantics web services discovery. In: Dillon, T.S., Chang, E., Meersman, R., Sycara, K. (eds.) Advances in Web Semantics I. LNCS, vol. 4891, pp. 346–373. Springer, Heidelberg (2009)