# Latent Semantic Indexing for Web Service Retrieval

Adam Czyszczoń and Aleksander Zgrzywa

Wrocław University of Technology, Division of Computer Science and Management, Institute of Informatics. Wybrzeże Wyspiańskiego 27, PL50370 Wrocław, Poland
`{adam.czyszczon,aleksander.zgrzywa}@pwr.edu.pl`
`http://www.zsi.ii.pwr.edu.pl/`

**Abstract.** This paper presents a novel approach for Web Service Retrieval that utilizes Latent Semantic Indexing method to index both SOAP and RESTful Web Services. Presented approach uses modified term-document matrix that allows to store scores for different service components separately. Service data is collected and extracted using web crawlers. To determine similarities between user query and services the cosine measure is used. Presented research results are compared to standard Latent Semantic Indexing method. We also introduce our Web Service test collection that can be used for many benchmarks and make research results comparable.

**Keywords:** Latent Semantic Indexing, Web Service Retrieval, Web Service.

## 1 Introduction

Web services are application components that are interoperable and platform independent. A large number of Web Services are being developed as building blocks to construct small or large-scale distributed Web Systems. There are two classes of Web Services – commonly referred to in the literature as SOAP Web Services and RESTful Web Services [1,2,3,4]. Services of the first class are used mainly in the industry. Their description is provided in WSDL (Web Service Description Language) documents and they exchange information using SOAP (Simple Object Access Protocol) protocol. RESTful Web Services are used mainly in Web applications. Their description is provided in HTML documents, can be and most frequently use JSON (JavaScript Object Notation) format to transfer data objects. Developers have access to a variety of services that are published on the Internet and some of the services provide similar functionality. Finding most suitable services from the vast collection available on the Web is still the key problem for service–oriented systems. Moreover, category-based Web Service Discovery methods were inefficient and ineffective as they relied on keyword matching.

In order to solve this problem, Platzer et al. [5] presented a Web Service Retrieval approach that used Information Retrieval (IR) methodology and utilized the Vector Space Model (VSM). This method, however, does not account for the order and association between terms [6]. To capture the higher-order association between terms and services, another IR approach can be applied – Latent Semantic Indexing (LSI). It also allows to find hidden semantic association between terms even though they do not appear together in any service. Such a higher-order association cannot be acquired by the VSM model or by keyword-based service discovery mechanism. It also allows to reduce the VSM term-document matrix by transforming it to matrices that represent the underlying structure.

In this paper we present a novel approach for Web Service Retrieval that utilizes LSI to index both SOAP and RESTful Web Services. Presented approach uses modified term-document matrix that allows to store scores for different service components separately. Service data is collected and extracted using web crawlers. Finally, we utilize the cosine measure to determine similarities between user query and services and to retrieve the corresponding relevant services sorted according to highest relevance. Presented research results concern retrieval effectiveness and are compared to standard LSI method. Additionally, we introduce and publish our Web Service test collection that can be used for many benchmarks and make research results comparable.

## 2 Related Work

Currently, two approaches to finding Web Services can be distinguished. The first one is Service Discovery [7] which concerned with matchmaking SOAP services assigned to prespecified categories in UDDI (Universal Description Discovery and Integration) repository. Second one, considered in this paper, is Web Service Retrieval which relies on Information Retrieval models and uses web crawlers to collect services that are publicly accessible on the Internet. In contrast to service discovery it can be applied to both SOAP and RESTful Web Services.

One of the earliest papers on LSI for Web Services was proposed in [6] where authors presented Service Discovery mechanism that retrieves services from UDDI registry based on concepts that were extracted from user query and retrieved from ontology framework. Descriptions of relevant services were later indexed into TF-IDF inverted index file (term-document matrix) and reduced using LSI method. Although the research presented the Service Discovery approach, its further drawback was indexing service descriptions only and concept categorization in UDDI. Despite its considerably high effectiveness, the ad-hoc indexing and operations concerned with ontology browsing put the performance of presented approach in question.

In [8] authors presented service retrieval solution using an approach that does not rely on ontology engineering. Presented approach utilized web crawling to collect WSDL documents, TF-IDF inverted index and LSI models. However, more information than just service description was indexed – authors also included operations, messages and data types. On the other hand, words occurring in selected service parameters were put together as one bag-of-words. Moreover, no effectiveness evaluation was presented. In further research [9] authors presented effectiveness evaluation but only for single term queries. The research also concerned only SOAP Web Services.

Another drawback of mentioned studies (and in Web Service Retrieval in general) is lack of common evaluation test collection of services for comparing different approaches. In many cases authors give only the information about destination URLs of public Web Service directories. Such a information becomes outdated as there appear new services, any of service providers makes changes to the service, or service becomes unavailable. Such a snapshot of a directory that is not published to others makes it impossible to compare the results on the same dataset.

## 3 Latent Semantic Indexing and Vector Space Model

Latent Semantic Indexing is a method for revealing hidden concepts in document data. This is achieved by finding the higher-order association between different keywords using the the Singular Value Decomposition (SVD). However, LSI is based on the VSM where documents are represented as a vectors. Every element in the document vector is calculated as weight that reflects the importance of a particular term that occurs in this document. In order to compute the weights, the *TF-IDF* (Term Frequency–Inverse Document Frequency) scheme is used. The file structure that allows to store such a data is called inverted index (hereinafter referred to as the index). This index is a term-document matrix where document vectors represent the columns and term vectors represent rows. User queries are also converted vectors in the same manner as documents. In order 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. The cosine value is referred to as the similarity degree and it is used for ranking documents relevance against user query. Based on the above we define VSM index as follows:

**Definition 1.** *The VSM index $A$ is a $m \times n$ matrix where $m$ denotes the number of terms in documents collection and $n$ represents collection's size. Every $a_{ij} \in A$ represents weight of $i-th$ term in $j-th$ document calculated using the TF-IDF scheme.*

The LSI is a variant of the VSM in which the original VSM matrix is replaced by a low-rank approximation matrix. Therefore, the original vector space is reduced to a sub-space as close as possible to the original matrix [9] using the Singular Value Decomposition. The result of the decomposition is following:

$$A = U\Sigma V^T \tag{1}$$

where $A$ is the initial VSM term($t$)-document($d$) matrix, $U \in R^{t \times t}$ is the matrix which columns represent term vectors, $\Sigma$ is a diagonal matrix of size $R^{t \times d}$ containing singular values, and $V \in R^{d \times d}$ is the matrix which columns represent the document vectors. Both $U$ and $V$ are orthogonal [10]. The number of singular values in $\Sigma$ determines the dimensions of the vector space. As such, we can reduce matrix $\Sigma$ into $\Sigma_K$ as $K \times K$ matrix to containing only the K singular values. The projection of $\Sigma_K$ on term matrix $U$ and document matrix $V^T$ allows to reduce them into $S_K$ and $U_K$ with K columns and rows. In result, initial matrix A is now approximated by:

$$A_K = U_K \Sigma_K V_K^T \tag{2}$$

After reduction, the terms are represented by the row vectors of the $m \times K$ matrix $U_K \Sigma_K$, and documents are represented by the column vectors the of the $K \times n$ matrix $\Sigma_K V_K^T$.

## 4 Web Service Structure

The definition of Web Service structure was elaborated in our previous study [11]. We consider Web Service to be composed of quadruple of elements where the first three represent *parameters* which correspond to service *name*, *description* and *version*, and the fourth represents service *components* which are composed of six-tuple of following component elements: *name*, *input value*, *output value*, *description.* The biggest advantage of presented structure is that it conforms to both SOAP and RESTful Web Services. The definition is as follows:

**Definition 2.** *$WS = \langle p_1, p_2, p_3, C \rangle$ where $p_1, p_2, p_3$ are service parameters and parameter $C$ denotes service component set $C = \{c_1, c_2, \ldots, c_n\}$ where each component $c_i \in C$ is represented by following four-tuple $c_i = \langle A_1, A_2, A_3, A_4 \rangle$.*

## 5 LSI for Web Service Retrieval

Current LSI approaches treat all service components as single bag-of-words. Based on our research carried out in [11] Web Services are indexed as in the following manner:

**Definition 3.** *The extended index $A$ is a $m \times n$ matrix where $m$ denotes the number of terms in Web Service collection and $n$ represents collection's size. Elements $a_{ij}$ represent service parameters a five-tuple $< p_1,\ p_2,\ p_3, c >$ where $p_1$ represents weights of service's name, $p_2$ service's description, $p_3$ service's version, and $c$ represents weight of all service components. Weights are calculated using the modified TF-IDF scheme presented in Equation 3.*

For such a index, we model Web Services in vector space in following manner: each parameter and component is represented as a vector composed of weights where each weight corresponds to a term from the "bag of words" of particular service parameter or component. The bag-of-words content of all service components is merged into single content. Weights are calculated using modification of one of the best known combination of *TF-IDF*:

$$TFIDF = (1 + log(tf(t, \alpha))) \cdot log \frac{N_\alpha}{|\{\alpha \in WS : t \in \alpha\}|} \tag{3}$$

where $tf(t, \alpha)$ is term frequency of term $t$ in parameter or component $\alpha$, $N_\alpha$ is the total number of $\alpha$ parameter/component, and $|\{\alpha \in WS : t \in \alpha\}|$ is the number of parameters/components where the term $t$ appears.

Because parameter/component counts are stored separately there is substantial difference in weights between standard LSI indexing model and the extended one. For example: there are 2 services $A$ and $B$. In service $A$ term $X$ occurs 1 time in name and 2 times in components. In service $B$ term $X$ does not occur at all. In basic index the $TF = 3$, $DF = 1$, and thus $TF-IDF = 0.44$. However, in extended index the *TF-IDF* is equal to 0.39 and $MWV = 0.01$, since for name the $TF = 1$, $DF = 1$, $TF - IDF = 0$, and for components $TF = 2$, $DF = 1$, $TF - IDF = 0.39$.

As we can see the extended VSM index is 4 times bigger than the standard one. In order to reduce it to its original size, we merge parameters/components using the *MWV* method presented in [11] and computed as average weight of all service parameters. The final index structure is following:

**Definition 4.** *The MVW index $A'$ is a $m \times n$ matrix where $m$ denotes the number of terms in Web Service collection and $n$ represents collection's size. Elements $a_{ij} \in A'$ represent the MWV weight of five-tuple $b_{ij} =< p_1, p_2, p_3, c >$ where $b_{ij} \in A$.*

Afterwards, for such a index the LSI model is applied.

## 6 Test Collections

In many studies on Web Service Retrieval and Discovery authors do not share their evaluation test collections of services. Instead, authors give only the information about destination URLs of public Web Service directories. Such a information quickly becomes outdated as new services appear, get modified or become unavailable. In result, comparing research outcome of different approaches is impractical. In order to solve this problem we prepared two test collections of SOAP Web Services[1]: *SOAP_WS_12*, *SOAP_WS_14*.

First collection contains 267 Web Services collected from *xmethods.net* – a directory of publicly available Web Services, used by many researchers for service retrieval benchmarks. The crawl was performed in 2012 for the purpose of the research presented in [11]. Second collection contains 662 services collected from: *service-repository.com*, *webservicex.net*, *venus.eas.asu.edu*, *visualwebservice.com* and *programmableweb.com* – popular and constantly updated directory of public Web Services.

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

At present, introduced in this paper 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 paper.

In Table 1 we present summary data of the above datasets. Collection named *SOAP_WS_12* will be used in this paper for retrieval effectiveness analysis and the second one for indexing performance analysis.

**Table 1.** Test collections structure

| | SOAP_WS_12 | SOAP_WS_14 |
|---|---|---|
| Services | 267 | 662 |
| Parameters | 432 | 886 |
| Components | 5140 | 18353 |
| Total elements | 5572 | 19239 |

## 7 Evaluation

Based on the approach presented in this paper we implemented indexing system that allowed us to conduct evaluation experiments. The goal of the experiment was to measure the performance and effectiveness of proposed approach. Additionally, we compared our results to standard Latent Semantic Indexing and Vector Space Model indexing methods. In order to evaluate the effectiveness we used the *WS_SOAP_12* test collection and the following classical information retrieval measures: *Precision*, *Recall*, *F-measure* ($\beta = 1$) and *Mean Average Precision (MAP)*. In order to evaluate performance we checked the indexing time

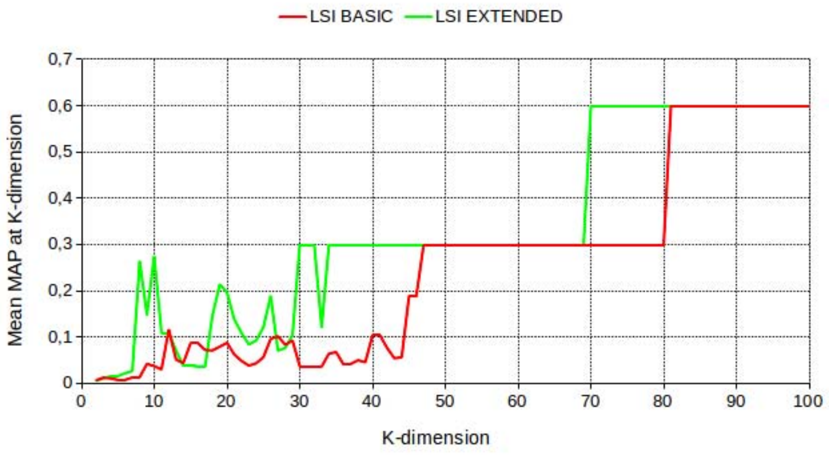

**Fig. 1.** Mean MAP at K-dimensions for basic and extended LSI

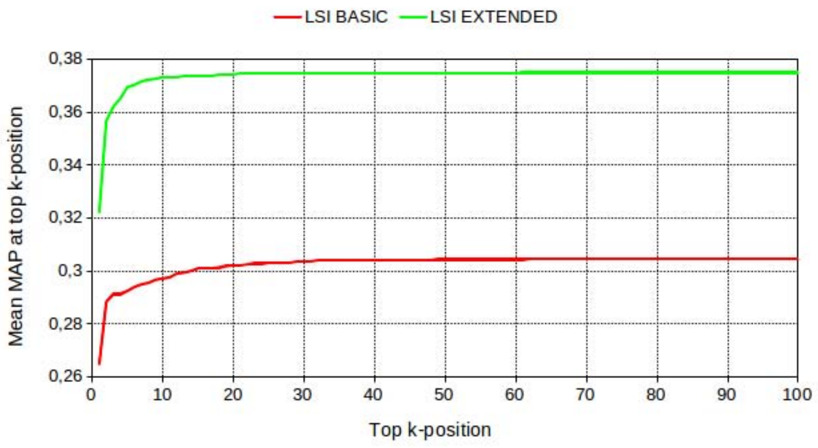

**Fig. 2.** Mean MAP at top $k$-positions for basic and extended LSI

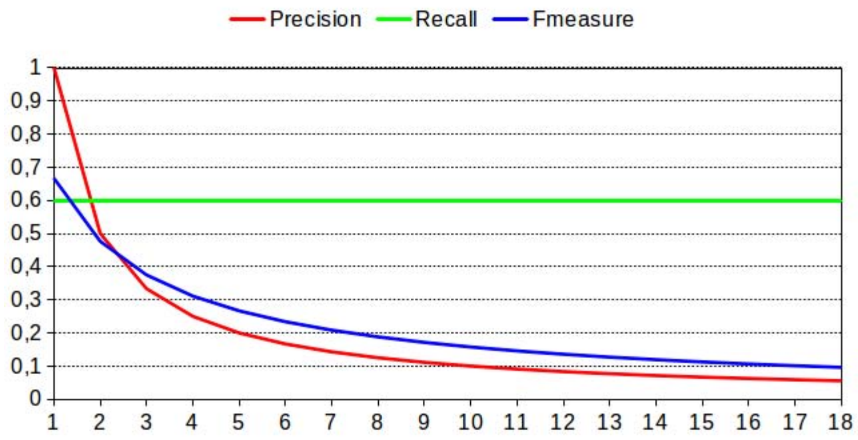

**Fig. 3.** Effectivenes evaluation for basic and extended LSI with K=81 dimensions

and memory consumption 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"*, denoted as $q_1$, $q_2$, ..., $q_8$. The MAP is computed as the average precision for queries $q_{1..8}$.

### 7.1 Dimension Reduction Analysis

The first step was to define the threshold for dimension reduction of the LSI index. One of the common approaches is to reduce the space to $K$ largest singular values [6]. However, the best $K$ value needs to be found empirically [9]. To do that, for every index structure with $K = [2, 267]$, we calculated the Mean of MAP of every top $k$ positions, where $k = [1, 267]$. This allowed us to check which value of $K$ will give the best retrieval results on top of the results list. The dataset used was the *WS_SOAP_12* with 267 Web Services. The experiment was conveyed for basic and extended LSI matrices. The results are presented on Figure 1.

In basic index the Mean MAP achieved its highest value equal to 0.6 at $K = 81$. Additionally, if we look at the MAP results for LSI index with $K = 81$, we can see that the highest MAP was achieved at first top position (see Figure 3). In the case of extended index, the highest value of Mean MAP was was also achieved at first top position but it was reached at smaller index with $K = 70$. This assures the same effectiveness as in the basic size but with 13.6% smaller index.

In Figure 2 we illustrate Mean MAP of top $k$-positions, for every $K$. Illustration confirm that the extended LSI reaches maximal recall faster, but also shows that it returns more relevant services at higher top positions. Most of the relevant services were returned within top 10 results.

### 7.2 Effectiveness Evaluation

In Figure 3 we illustrate the retrieval effectivenes of both index structures using LSI index with $K = 81$ dimenstions at top 20 positions. Relevant services were found at the very top of the list. However, not all relevant services were returned (recall=0.6). The overall effectiveness of LSI indicate that proposed meyhod performs very well and results are satisfactory.

## 8 Conclusions and Future Work

The goal of presented research was to propose an alternative LSI approach for Web Service Retrieval. The modified LSI approach included modified matrix that allowed to count scores for different service components separately. In result, the overall retrieval effectiveness should be higher than in the basic LSI index.

The experimental results proved that proposed extended model of LSI is superior to the standard model in terms of indexing performance and retrieval effectiveness. The dimension reduction showed not only the optimal index size values for basic and extended methods, but also confirmed that extended index achieved its maximal effectiveness at 13.6% smaller index size. Additionally, the extended index returned more relevant services within smaller list of top results.

In further research we plan to check effectiveness of different indexing methods and *TF-IDF* variants, especially the *ltc.lnc* in SMART notation, since the calculation of *IDF* for queries requires additional computation. This, in result, requires to load additional data into computer memory. Some researchers on Web Service Retrieval also suggested that skipping length-normalization may bring better results. This is also a subject of our future work on Web Service Retrieval methods.

## References

1. Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., Orchard, D.: Web services architecture. W3C Working Group Note World Wide Web Consortium (February 11, 2004), `http://www.w3.org/TR/ws-arch` (accessed: April 2014)
2. 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. In: The Prentice Hall Service-oriented Computing Series from Thomas Erl. Prentice Hall (2011)
3. 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)
4. Richardson, L., Ruby, S.: RESTful Web Services: Web Services for the Real World. O'Reilly Media, Inc., Sebastopol (2007)
5. 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)
6. Paliwal, A.V., Adam, N.R., Bornhovd, C.: Web service discovery: Adding semantics through service request expansion and latent semantic indexing. In: IEEE SCC, pp. 106–113. IEEE Computer Society (2007)
7. Mukhopadhyay, D., Chougule, A.: A survey on web service discovery approaches. CoRR abs/1206.5582 (2012)
8. 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)
9. Wu, C., Potdar, V., Chang, E.: Latent semantic analysis – the dynamics of semantics web services discovery. In: Dillon, T.S., et al. (eds.) Advances in Web Semantics I, vol. 4891, pp. 346–373. Springer, Heidelberg (2009)
10. Hyung, Z., Lee, K., Lee, K.: Music recommendation using text analysis on song requests to radio stations. Expert Systems with Applications 41(5), 2608–2618 (2014)
11. 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)