Multilabel Learning for Automatic Web Services Tagging

Mustapha AZNAG
Aix-Marseille University,
LSIS UMR 7296, France.
mustapha.aznag@univ-amu.fr

Mohamed QUAFAFOU
Aix-Marseille University,
LSIS UMR 7296, France.
mohamed.quafafou@univ-amu.fr

Zahi JARIR
University of Cadi Ayyad Marrakech,
LISI Laboratory, FSSM, Morocco.
jarir@uca.ma

***Abstract*—Recently, some web services portals and search engines as Biocatalogue and Seekda!, have allowed users to manually annotate Web services using tags. User Tags provide meaningful descriptions of services and allow users to index and organize their contents. Tagging technique is widely used to annotate objects in Web 2.0 applications. In this paper we propose a novel probabilistic topic model (which extends the CorrLDA model - Correspondence Latent Dirichlet Allocation-) to automatically tag web services according to existing manual tags. Our probabilistic topic model is a latent variable model that exploits local correlation labels. Indeed, exploiting label correlations is a challenging and crucial problem especially in multi-label learning context. Moreover, several existing systems can recommend tags for web services based on existing manual tags. In most cases, the manual tags have better quality. We also develop three strategies to automatically recommend the best tags for web services. We also propose, in this paper, WS-Portal; An Enriched Web Services Search Engine which contains 7063 providers, 115 sub-classes of category and 22236 web services crawled from the Internet. In WS-Portal, severals technologies are employed to improve the effectiveness of web service discovery (i.e. web services clustering, tags recommendation, services rating and monitoring). Our experiments are performed out based on real-world web services. The comparisons of Precision@n, Normalised Discounted Cumulative Gain (NDCGn) values for our approach indicate that the method presented in this paper outperforms the method based on the CorrLDA in terms of ranking and quality of generated tags.**

Keywords*—*Web services, Tags, Automatic, Recommendation, Machine Learning, Topic Models.

I. Introduction

The Service Oriented Architecture (SOA) is a model currently used to provide services on the Internet. The SOA follows the find-bind-execute paradigm in which service providers register their services in public or private registries, which clients use to locate web services. Web services[1] [27] are defined as software systems designed to support interoperable machine-to-machine interaction over a network. They are loosely coupled reusable software components that encapsulate discrete functionality and are distributed and programmatically accessible over the Internet. They are self contained, modular business applications that have open, internet-oriented and standards based interfaces [1]. Web services are autonomous software components widely used in various SOA applications according to their platform-independent nature. Different tasks like matching, ranking, discovery and composition have been intensively studied to improve the general web services management process. Thus, the web services community has proposed different approaches and methods to deal with these tasks.

Recently, some web services portals and search engines as Biocatalogue[2] and Seekda![3] (Currently, the portal is no longer available.) and some other web services portals also support tags, have allowed users to manually annotate Web services using tags. User Tags provide meaningful descriptions of services and allow users to index and organize their contents. Tagging technique is widely used to annotate objects in Web 2.0 applications. This type of metadata provides a brief description of Web services and allows users to find appropriate services more easily. Tagging data provides meaningful descriptions, and is utilized as another information source for Web services.

Several web services tagging approaches have been proposed, for example the tagging system proposed in [14], [20]. However, most of them annotate web services manually. Moreover, several existing systems can recommend tags for web services based on existing manual tags [13], [9]. In most cases, the manual tags have better quality. In this paper we propose a novel approach based on our previous work on probabilistic topic models [23] to automatically tag web services according to existing manual tags. Our probabilistic topic model is a latent variable model that exploits local correlation labels. Indeed, exploiting label correlations is a challenging and crucial problem especially in Multi-Label learning context. We also develop three strategies to automatically recommend the best tags for web services. Our experiments are performed out based on real-world web services (i.e. Section IV). The experiment results show that the performance of our approach is affected by web services with or without user's tags. For this, we propose three strategies to learn the classifier before recommendation task.

The main contributions of this paper can be summarized as follows:

1) We propose an automatic tagging technique for web services, in which both the WSDL documents and service tags are effectively utilized. Our approach can work without existing tags, and works better when there exists manual tags.
2) We propose three tag recommendation strategies to improve the performance of our approach. We exploit

[1] http://www.w3.org/standards/webofservices

[2] https://www.biocatalogue.org/

[3] http://webservices.seekda.com/

WSDL documents and related descriptions to extract the most important words and user's tags.

3) We generate tags for 22,236 real web services and these tags are published online in our developed Web Services Portal[4]

To validate the performance of our approach, a series of experiments are carried out. The comparisons of Precision@n, Normalised Discounted Cumulative Gain (NDCGn) values for our approach indicate that the method presented in this paper outperforms better when the selected tags from WSDL description are combined with the existing manual tags.

In this paper we propose also an enriched web service search engine called WS-Portal[4] where we incorporate our research works to facilitate web services discovery task (see Section V) [6].

The rest of this paper is organized as follows. Section II analyzes some related work. In Section III, we describe in detail our web services tag recommendation approach. Section IV describes the experimental evaluation. Section V describes our developed web services search engine. Finally, the conclusion and future work can be found in Section VI.

II. Related work

Generally, every web service associates with a WSDL document that contains the description of the service. A lot of research efforts have been devoted in utilizing WSDL documents and Web service clustering [28], [19], [18], [12], [11] has been demonstrated as an effective mechanism to boost the performance of Web services discovery. Dong et al. [11] proposed the Web services search engine Woogle that is capable of providing Web services similarity search. However, their engine does not adequately consider data types, which usually reveal important information about the functionalities of Web services [18]. Liu and Wong [19] apply text mining techniques to extract features such as service content, context, host name, and service name, from Web service description files in order to cluster Web services. They proposed an integrated feature mining and clustering approach for Web services as a predecessor to discovery, hoping to help in building a search engine to crawl and cluster non-semantic Web services. Elgazzar et al. [12] proposed a similar approach which clusters WSDL documents to improve the non-semantic web service discovery. They take the elements in WSDL documents as their feature, and cluster web services into functionality based clusters. The clustering results can be used to improve the quality of web service search results.

Recently, tagging data provides meaningful descriptions, and is utilized as another information source for Web service. In this section, we briefly discuss some existing research works of tagging data related to different problems in web service. Meyer et al. use tags to annotate web services semantically [20]. Similary this idea, Gawinecki et al. use structured collaborative tags to matchmake web services [14]. However, all these tags are generated manually and the authors spend 12 days to generate tags for just 50 services. Thus, manual tagging is very time-consuming and an automatic tagging system is needed for web services. To handle the problem of limited tags, Azmeh et al. [2] propose an automatic tagging system for web services which extracts tags from WSDL documents using machine learning technology and WordNet synsets. The system uses relevant synonyms in WordNet to enrich tags. Fang et al. [13] propose an approach to generate tags for web services automatically using two tagging strategies, tag enriching and tag extraction. In the first strategy, the system use clustering technique to enrich tags with existing manual tags. In the second strategy, recommended tags are extracted from WSDL documents and related descriptions. Liang et al. [10] propose a hybrid mechanism by using service-tag network information to compute the relevance scores of tags by employing semantic computation and HITS model, respectively.

In [9], the authors improve the performance of Web service clustering by introducing a novel approach based on the Author-Topic-Model [24] to explore the knowledge behind WSDL documents and tags and by proposing three tag preprocessing strategies to improve the performance of service clustering. But the system can't work if there is no manual tag in the system. Topic models are successfully used for a wide variety of applications including documents clustering and information retrieval [26], collaborative filtering [15], and visualization [16] as well as for modeling annotated data [8]. In our previous work [3], [4], we investigated the use of three probabilistic topic models PLSA, LDA and CTM to extract topics from semantically enriched service descriptions. These topics provide a model which represents any web service's description by a vector of terms. In our approach, we assumed all service descriptions were written in the WSDL and/or SAWSDL. The results obtained from comparing the three methods based on PLSA, LDA and CTM showed that the CTM model provides a scalable and interoperable solution for automated service discovery and ranking in large service repositories. The CTM model assumes that the concepts of each service arise from a mixture of topics, each of which is a distribution over the vocabulary. In this paper, we use CTM model to extract and select the candidates tag for a web services in the dataset. Then, we use the extracted tags from web service dataset to train our classifier using a latent variable model based on LocLDA (Local Correspondence Latent Dirichlet Allocation), which is a latent variable model that exploits local correlation labels [23]. LocLDA was built on Correspondence Latent Dirichlet Allocation (Corr-LDA) [8].

III. Web Services Tags Recommendation System

In this section, we describe the details of our web services tags recommendation approach. The overall process of our approach is divided into three phases:

1) Web Services Representation and Tags Extraction: We process the service descriptions and we use a probabilistic method to extract and select the candidates tag for a web services in the dataset (Section III-A).
2) Training Web Services Tags Recommendation Classifier: We use the extracted tags from web services dataset to train our classifier using a latent variable model (Section III-B).

[4] WS-Portal is available online:

- http://wvmweb.esil.univ-mrs.fr/wsportal
- http://www.webvirtualmachine.fr/wsportal
- http://wsportal.aznag.net

3) Web Services Tags Recommendation: Finally, we use the trained classifier to recommend the best tags for a new web service (Section III-C).

A. *Web Services Representation and Tags Extraction*

Web services are generally described with a standard Web Service Description Language (WSDL). The WSDL is an XML-based language, designed according to standards specified by the W3C, that provides a model for describing web services. It provides the specifications necessary to use the web service by describing the communication protocol, the message format required to communicate with the service, the operations that the client can invoke and the service location. To manage efficiently web service descriptions, we extract all features that describe a web service from the WSDL document (i.e. such as services, documentation, messages, types and operations).

As shown in Figure 1, our tags extraction process contains two main components, features extraction and tags selection. Before representing web services as TF-IDF (Text Frequency and Inverse Frequency) [25] vectors, we need some preprocessing. There are commonly several steps:

- *Features extraction* extracts all features that describe a web service from the WSDL document, such as service name and documentation, messages, types and operations.
- *Tokenization:* Some terms are composed by several words, which is a combination of simple terms (*e.g., get_ComedyFilm_MaxPrice_Quality*). We use therefore regular expression to extract these simple terms (*e.g., get, Comedy, Film, Max, Price, Quality*).
- *Stop words removal*: This step removes all HTML tags, CSS components, symbols (punctuation, etc.) and stop words, such as 'a', 'what', etc. The Standford POS Tagger[5] is then used to eliminate all the tags and stop words and only words tagged as nouns, verbs and adjectives are retained. We also remove the WSDL specific stop words, such as *host, url, http, ftp, soap, type, binding, endpoint, get, set, request, response*, etc.
- *Word stemming*: We need to stem the words to their origins, which means that we only consider the root form of words. In this step we use the Porter Stemmer Algorithm [22] to remove words which have the same stem. Words with the same stem will usually have the same meaning. For example, 'computer', 'computing' and 'compute' have the stem 'comput'. The Stemming process is more effective to identify the correlation between web services by representing them using these common stems (root forms).

After identifying all the functional terms, we calculate the frequency of these terms for all web services. We use the Vector Space Model (VSM) technique to represent each web service as a vector of these terms. In fact, it converts service description to vector form in order to facilitate the computational analysis of data. In information retrieval, VSM is identified as the most widely used representation for documents and is a very useful method for analyzing service descriptions. The TF-IDF algorithm [25] is used to represent a dataset of WSDL documents and convert it to VSM form. We use this technique, to represent a services descriptions in the form of *Service Transaction Matrix (STM)*. In STM, each row represents a WSDL service description, each column represents a word from the whole text corpus (vocabulary) and each entry represents the TF-IDF weight of a word appearing in a WSDL document. TF-IDF gives a weight w_{ij} to every term j in a service description i using the following equation:

$$w_{ij} = tf_{ij}.\log(\frac{n}{n_j}) \qquad (1)$$

Where tf_{ij} is the frequency of term j in WSDL document i, n is the total number of WSDL documents in the dataset, and n_j is the number of services that contain term j.

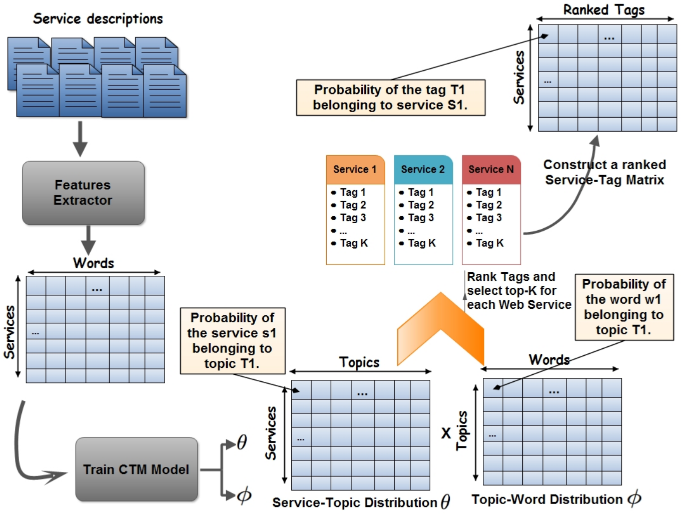

Service descriptions
Features Extractor
Words
Services
...
Train CTM Model
θ
ϕ
Probability of the service s1 belonging to topic T1.
Topics
Services
...
X
Topics
Words
...
Service-Topic Distribution θ
Topic-Word Distribution ϕ
Probability of the word w1 belonging to topic T1.
Rank Tags and select top-K for each Web Service
Service 1
Service 2
Service N
• Tag 1
• Tag 2
• Tag 3
• ...
• Tag K
Construct a ranked Service-Tag Matrix
Ranked Tags
Services
Probability of the tag T1 belonging to service S1.

Fig. 1. An Overview of Web Services Tags Extraction Mechanism

Figure 1 presents an overview of our proposed Web Service Tag Extraction mechanism. For each Web Service, we generate top-K tags using our previous approach based on Correlated Topic Model (CTM) described in [3]. We utilized CTM to extract latent factors $z_f \in Z = \{z_1, z_2, ..., z_k\}$ from web service descriptions (i.e., *STM*). In our work we use STM as training data for our implementation of CTM model. After the CTM model is trained, the distribution of words for each topic is known and all the services in the dataset can be described as a distribution of topics. Let

1) $\theta^{(s)} = P(z)$ refer to the multinomial distribution over topics in the service description s.
2) $\phi^{(j)} = P(w|z_j)$ refer to the multinomial distribution over words for the topic z_j.

Then, we use the extracted topics to rank the related tags for each web service. Each tag w in a service description s is generated by sampling a topic z from topic distribution (i.e. ϕ), and then sampling a word from topic-word distribution (i.e. θ). The probability of the ith tag occurring in a given service is given by Equation 2:

[5] http://nlp.stanford.edu/software/tagger.shtml

$$P(t_i|s) = \sum_{f=1}^{k} P(t_i|z_f)P(z_f|s) \qquad (2)$$

Where z_f is a topic from which the *i*th word was drawn, $P(z_f|s)$ is the probability of topic z_f in the service s, and $P(t_i|z_f)$ is the probability of having tag t_i given the fth topic. The most relevant tags are the ones that maximize the probability $P(t_i|s)$ for a service s (See Algorithm 1)

Finally, we represent the output of this step by a matrix that contains for each web service the most related tags ranked in the descending order (i.e. $P(t_i|s)$). The main key of our approach is that the selected tags for a web service are not necessarily in its descriptions. In fact, we have represented all services in a topic space and tags are related to these topics. The result of this step will be used as input for the training classifier phase (Section III-B).

Algorithm 1 Web services tags extraction

```
Require: • $S = \{s_1, \ldots, s_D\}$ web services set. ($D$ number of services).
         • K Number of Topics.
Ensure: Ranked tags for each service.
Perform CTM on services set $S = \{s_1, \ldots, s_D\}$.
for each service $s_i \in S = \{s_1, \ldots, s_D\}$ do
    for each word $w_m, m \in \{1, ..., M\}$ do
        Compute $P(w_m|s_i)$ (Equation 2)
    end for
    RankTags: The most relevant words are the ones that maximize the probability $P(w_m|s_i)$.
end for
return Set of $K$ ranked tags for each service.
```

Symbol	Description
D	Number of service in training set.
K	Number of topics.
M	Number of words related to a service.
T	Number of tags.
V	Neighborhood tags.
v	A neighbor tag: $v \in Index(Parents(tag))$.
θ	Multinomial distribution over topics: θ_i, $i \in \{1, ..., K\}$
z	Latent topic. $z_m^i = 1$ if z_m is the *i*th latent topic, else $z_m^i = 0$.
w	Word. $w_m^j = 1$ if w_m is the *j*th word else $w_m^j = 0$.
t	Tag. $t_n^j = 1$ if t_n is the *j*th tag else $t_n^j = 0$.
y	Discrete indexing variable.
W_s	Size of words vocaculary.
W_t	Size of tags vocabulary.
α	Dirichlet prior for θ: α_i, $i \in \{1, ..., K\}$
π	Multinomial: π_{ij}, $i \in \{1, ..., K\}$, $j \in \{1, ..., W_s\}$
β	Multinomial: β_{ij}, $i \in \{1, ..., K\}$, $j \in \{1, ..., W_t\}$
ϕ	Variational Multinomial: ϕ_{mi}, $m \in \{1, ..., M\}$, $i \in \{1, ..., K\}$
γ	Variational Dirichlet: γ_i, $i \in \{1, ..., K\}$
λ	Variational Multinomial: λ_{nm}, $n \in \{1, ..., T\}$, $m \in \{1, ..., M\}$
ψ	The digamma function, the first derivative of the log Gamma function.

TABLE I. NOTATIONS USED IN THIS PAPER

B. Training Web Services Tags Recommendation Classifier

In this step we have a dataset of service descriptions and extracted tags. From this training dataset, we first extract a list of candidate words using the probabilistic method based on CTM (Section III-A). Using this set of candidate tags, service transaction matrix and the original tags (manual tags) we train a classifier. We define our tags recommendation task as follows: given a set of web services, in which each service has not only a bag of words but also a bag of tags, our task is to learn a model using this dataset; and, when given an unseen service in which only the content words can be observed, we should predict a ranked list of tags based on the learned model and the observed words in the service. Our probabilistic approach based on LocLDA model (Local Correspondence Latent Dirichlet Allocation), which is a latent variable model that exploits local correlation labels [23]. LocLDA was built on Correspondence Latent Dirichlet Allocation (CorrLDA) [8]. More precisely, our model calculates dynamically the model structure depending on the data, and particularly, on the interaction between annotations. We originally developed the LocLDA model for generating captions for images. An image contains multiple regions, and each word in the image caption corresponds to one of the regions. The correspondence from words to regions is assumed to follow uniform distributions [23]. For our tags recommendation task, we adopt the LocLDA model for modeling the correspondence from the topic assignments for words and the topic assignments for tags of web services. We apply this model to the cases of automatic web services tagging. Given a service s with no tags, the task is to predict its missing tags.

Let $z = \{z_1, z_2, ..., z_K\}$ be the latent factors that generate the web service, and $y = \{y_1, y_2, ..., y_T\}$ be discrete indexing variables that take values from *1* to *T* with equal probability. Table I shows the notations used in this paper. Conditioned on *T* (i.e. Number of tags) and *M* (i.e. Number of words related to a web service), a *K*-topics (i.e. Number of topics), LocLDA model (Figure 2) assumes the following generative process for a pair service/tag (w, t):

1) Find the parents of each tag.
2) Sample $\theta \sim Dirichlet(\theta|\alpha)$
3) For each word $w_m, m \in \{1, ..., M\}$
   - Sample $z_m \sim Multinomial(\theta)$
   - Sample $w_m \sim p(w|z_m, \pi)$ from a multinomial distribution conditioned on z_m
4) For each tag $t_j, j \in \{1, ..., T\}$
   - Sample $y_j \sim Uniform(1, ..., T)$
   - Sample $t_j \sim p(t|y_j, y_v, \mathbf{z}, \beta)$

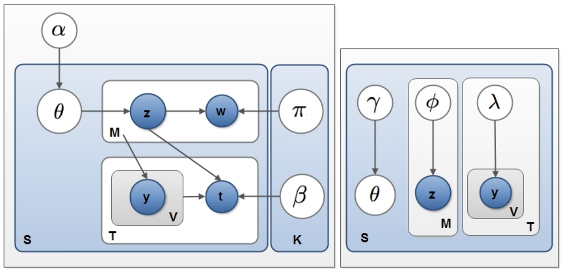


Fig. 2. (Left) Graphical model of LocLDA, (Right) Representation of variational distribution used to approximate the posterior in LocLDA.

In our model, the correspondence between a tag t_j and its associated service is obtained via a latent variable y_j. We consider that y_j is the parent of t_j and we note it by

$y_j = Parent(t_j)$. Thus, we would like to represent the interaction between different tags describing the same service. Let consider that the true caption of a given service $\mathcal{S}$ is $\mathcal{T} = \{t_1, t_2, ..., t_n\}$. We no longer consider that a tag $t_j \in \mathcal{T}$ is connected to the service $\mathcal{S}$ via a single latent variable, but through a set of latent variables y_j which are parents of $t_j \in \mathcal{T} - t_j\}$ tags. To obtain the parents of a given tag, we first determine its neighbors by performing a multiple regression on each tag with respect to all other tags. Indeed, multiple regression is a statistical analysis that describes the relationships among variables [23]. Given a set of tags $\{t_1, t_2, ..., t_n\}$, we seek to explain precisely the values taken by a single tag from all other tags. This process is performed for all tags. The theoretical model, formulated in terms of random variables, takes the form:

$$t_j = a_0 + a_1 t_1 + a_2 t_2 + ... + a_n t_n + \epsilon_j$$

where ϵ is the model error that expresses the missing information in the explanation values of t_j from t_{-j}. t_{-j} represents all the tags not including the *j*th one. $a_1, a_2, ..., a_n$ are parameters to be estimated. By setting a threshold for the parameters a_i, we obtain the neighbors of a tag. We use the notation *Index*, which gives the indices of the parents of each tag (i.e. $v \in Index(Parents(tag))$ where v is a neighbor tag).

LocLDA model defines the joint distribution of the service description, tags and topics as follows:

$$\begin{aligned} P(\mathbf{w}, \mathbf{t}, \theta, \mathbf{z}, \mathbf{y}|\alpha, \pi, \beta) = P(\theta|\alpha) \quad & \prod_{m=1}^{M} P(z_m|\theta) P(w_m|z_m, \pi)\Bigg) \\ & \prod_{j=1}^{T} \prod_{v} P(y_j|M) P(y_v|M) P(t_j|y_j, y_v, z, \beta)\Bigg) \end{aligned} \quad (3)$$

where α, π and β are the parameters to estimate.

The exact probabilistic inference is intractable for LocLDA, therefore, we turn to variational inference methods [17] to approximate the posterior distribution of the latent variables given a service/tag. We introduce a variational distribution q on the latent variables:

$$q(\theta, \mathbf{z}, \mathbf{y}) = q(\theta|\gamma) \quad \prod_{m=1}^{M} q(z_m|\phi_m)\Bigg) \Bigg(\prod_{n=1}^{N} q(y_n|\lambda_n) \prod_{v} q(y_v|\lambda_v)\Bigg) \quad (4)$$

where γ, ϕ and λ are variational parameters.

The objective is to optimize the values of the variational parameters that make the variational distribution q close to the true posterior p by minimizing the Kullback-Leibler (KL) divergence between the variational distribution and the true posterior. We bound the log-likelihood of a given service/tag using Jensen's inequality:

$$\begin{aligned} & L(\gamma, \phi, \lambda; \alpha, \pi, \beta) \\ & \quad = E_q[\log P(\theta|\alpha)] + E_q[\log P(\mathbf{z}|\theta)] + E_q[\log P(\mathbf{w}|\mathbf{z}, \pi)] \\ & \quad + E_q[\log P(\mathbf{y}|M)] + E_q[\log P(\mathbf{t}|\mathbf{y} \in parents(\mathbf{t}), z, \beta)] \\ & \quad - E_q[\log q(\theta|\gamma)] - E_q[\log q(\mathbf{z}|\phi)] - E_q[\log q(\mathbf{y}|\lambda)] \end{aligned} \quad (5)$$

Thus, by expanding each term of the equation 5 with respect to maximizing each variational parameter, we find the following updates rules:

1) Update the posterior Dirichlet parameters

$$\gamma_i = \alpha_i + \sum_{m=1}^{M} \phi_{mi} \quad (6)$$

2) For each service, update the posterior distribution over topics

$$\begin{aligned} \phi_{mi} \propto \pi_{iw_m} \exp\Bigg(& \psi(\gamma_i) - \psi(\sum_{j=1}^{K} \gamma_j) \\ & + \sum_{n=1}^{N} \sum_{v} \lambda_{nm} \lambda_{vm} \log \beta_{it_n}\Bigg) \end{aligned} \quad (7)$$

3) For each tag, update the posterior distribution over services

$$\lambda_{nm} \propto \exp \quad \sum_{i=1}^{K} \sum_{v} \phi_{mi} \lambda_{vm} \log \beta_{it_n}\Bigg) \quad (8)$$

We maximize the lower bound with respect to the model parameters α, π, β. Given a training services set $D = \{(w_d, t_d)\}_{d=1}^{D}$, the objective is to find the maximum likelihood estimation for α, π, β. The corpus log-likelihood is bounded by :

$$L(D) = \sum_{d=1}^{D} \log P(w_d, t_d|\alpha, \pi, \beta) \geq \sum_{d=1}^{D} L(\gamma_d, \phi_d, \lambda_d; \alpha, \pi, \beta)$$

We then find α, π, β that maximize this lower bound:

$$\pi_{ij} \propto \sum_{d=1}^{D} \sum_{m=1}^{M_d} \phi_{dmi} w_{dm}^{j} \quad (9)$$

$$\beta_{ij} \propto \sum_{d=1}^{D} \sum_{n=1}^{N_d} t_{dn}^{j} \sum_{m=1}^{M_d} \sum_{v} \phi_{dmi} \lambda_{dnm} \lambda_{dvm} \quad (10)$$

Finally, the Newton-Raphson algorithm [7] is used to estimate the Dirichlet α.

After obtaining parents of tags using the regression method, we present also in this section the variational EM algorithm [21] which performs iterative maximization of a lower bound of data in which some variables are unobserved. It maximizes a lower bound of the data log-likelihood with respect to the variational parameters, and then, for fixed values of the variational parameters, maximizes the lower bound with respect to the model parameters. Indeed, we have the following iterative algorithm:

- (E-Step) For each service, find the optimizing values of the variational parameters using equations (6), (7) and (8) with appropriate starting points for γ, ϕ_{mi} and λ_{nm}.
- (M-Step) Maximize the resulting lower bound on the log-likelihood with respect to the model parameters

for fixed values of the variational parameters, using equations (9), (10) and the Newton-Raphson algorithm.

These two steps are repeated until the lower bound on the log-likelihood converges.

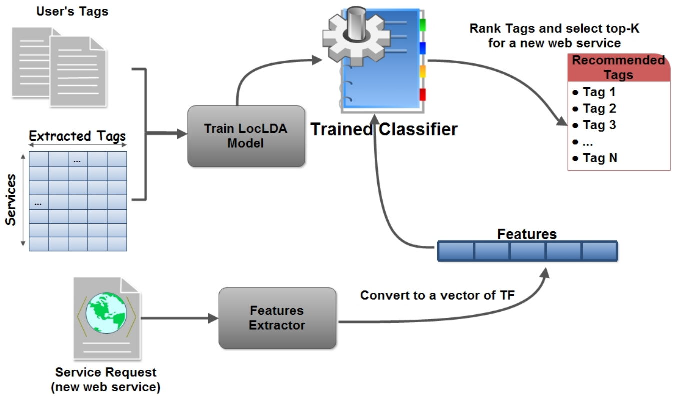


Fig. 3. An overview of Web Services Tags Recommendation mechanism

C. Web Services Tags Recommendation

Once our classifier has been trained, we can recommend tags for a new web service by performing the variational inference with fixed model parameters α, β and π. Thus, we can compute the conditional distributions of untagged service $p(t|s_{new})$ (Equation 11).

$$P(t|s_{new}) = \sum_{m=1}^{M} \sum_{z_m} P(z_m|\theta)P(t|z_m, \beta) \tag{11}$$

The most relevant tags are the ones that maximize the probability $P(t|s_{new})$ for a service s_{new} (See Algorithm 2)

Figure 3 presents an overview of our proposed Web Service Tag Recommendation mechanism.

Algorithm 2 Web services tags recommendation

```
Require: .
    • $\mathcal{S} = \{s_1, \ldots, s_D\}$ web services set. ($D$ number of services).
    • Set of extracted tags $\mathcal{E} = \{e_1, \ldots, e_M\}$ ($M$ number of extracted tags) (Algorithm 1).
    • Set of original tags (manual tags) $\mathcal{T} = \{t_1, \ldots, t_T\}$ ($T$ number of original tags).
    • K Number of Topics.
    • Given service $s_{new}$.
Ensure: $R$ Ranked tags for a given service.
Perform LocLDA on services datasets $\mathcal{S}$, $\mathcal{E}$ and $\mathcal{T}$
for each tag $t \in \mathcal{T}$ do
    Compute $P(t|s_{new})$ (Equation 11)
end for
RankTags: The most relevant tags are the ones that maximize the probability $P(t|s_{new})$.
return Set of $R$ ranked tags for a new web service $s_{new}$.
```

IV. Evaluation

A. Web Services Corpus

Our experiments are performed out based on real-world web services that we collected from the web since 2011. We have considered different web service sources like WebservicesX.net[6], xMethods.net[7], Seekda![8], Service-Finder![9] and Biocatalogue[10]. We have collected 22,236 real web services. For each Web service, we get the WSDL document and related tags if they exist. We generate tags for all web services in the dataset and these tags are published online in our Web Services search engine[4].

Before applying the proposed approach, we process the WSDL corpus. The objective of this pre-processing is to identify the textual words of services, which describe the semantics of their functionalities. WSDL corpus processing consists of several steps: *Features extraction*, *Tokenization:*, *Stop words removal*, *Word stemming* and *Service Transaction Matrix construction*. The observed words are represented in a Service Transaction Matrix (STM). In our work we use service transaction matrix as training data for our models.

To evaluate our method, we select 633 web services from our dataset. all these services have manual tags. We selected only the services having 3 to 10 manual tags, and there are totally 739 manual tags belonging to them. Then we use different approaches to tag these web services:

1) Original tags: In this approach, we just use the 633 web services and their tags to train our classifier and recommend tags for new web services.
2) Extracted tags: In this approach, we just generate extracted tags and select top-k extracted tags as final results (III-A).
3) Original tags + Extracted tags: In this approach, we mix original tags with extracted tags from WSDL documents.

The proposed approach is evaluated using the *Precision at n* ($Precision@n$) and the *Normalised Discounted Cumulative Gain* ($NDCG_n$) for the generated tags obtained for each of the service in the test set.

All experiments were performed on a Dell 64-bit Server with Intel®Xeon(R) CPU X5560 @ 2.80GHz x 16 and 16 Go of RAM.

B. Metrics Evaluation

In order to evaluate the accuracy of our approach, we compute two standard measures used in *Information Retrieval*: *Precision at n* (*Precision@n*) and *Normalised Discounted Cumulative Gain* ($NDCG_n$). $Precision@n$ and $NDCG_n$ are widely accepted as the metrics for ranking evaluation in IR. Formally, the previous metrics are defined as follows:

[6] http://www.webservicex.net/ws/default.aspx
[7] http://www.xmethods.net/ve2/index.po
[8] http://www.webservices.seekda.com
[9] http://demo.service-finder.eu/search
[10] https://www.biocatalogue.org/

*1) **Precision@n**:* In our context, $Precision@n$ is a measure of the precision of the service tag recommendation and ranking system taking into account the first n retrieved tags. The precision@n for a list of retrieved tags is given by Equation 12:

$$Precision@n = \frac{|RelevantTags \cap RetrievedTags|}{|RetrievedTags|} \tag{12}$$

Where the list of relevant tags to a given service is the ground truth tags related to the service.

*2) **Normalised Discounted Cumulative Gain**:* $NDCG_n$ uses a graded relevance scale of each retrieved tag from the result set to evaluate the gain, or usefulness, of a tag based on its position in the result list. This measure is particularly useful in Information Retrieval for evaluating ranking results. The $NDCG_n$ for n retrieved tags is given by Equation 13.

$$NDCG_n = \frac{DCG_n}{IDCG_n} \tag{13}$$

Where DCG_n is the Discounted Cumulative Gain and $IDCG_n$ is the Ideal Discounted Cumulative Gain. The $IDCG_n$ is found by calculating the DCG_n of the ideal first n generated tags for a given service. The DCG_n is given by Equation 14

$$DCG_n = \sum_{i=1}^{n} \frac{2^{relevance(i)} - 1}{log_2(1+i)} \tag{14}$$

Where n is the number of tags retrieved and $relevance(s)$ is the graded relevance of the tag in the ith position in the ranked list. The $NDCG_n$ values for all tags can be averaged to obtain a measure of the average performance of a ranking algorithm. $NDCG_n$ values vary from 0 to 1.

In Information retrieval, $NDCG_n$ gives higher scores to systems which rank a result list with higher relevance first and penalizes systems which return tags with low relevance.

*3) **Caption Perplexity**:* We compute the perplexity of the given tags under $P(t|s)$ for each service s in the test set to measure the tags quality of the models. In computational linguisticics, the measure of perplexity has been proposed to assess generalizability of text models. The perplexity is algebraically equivalent to the inverse of the geometric mean per-word likelihood [8]. A lower perplexity score indicates better generalization performance. Assume we have D web services as a held-out dataset D_{test} and each web service s contains N_d tags. More formally, the perplexity for a dataset D_{test} is defined by:

$$Perplexity = \exp \left. - \sum_{d=1}^{D} \sum_{n=1}^{N_d} \frac{\log P(t_n|s_d)}{\sum_{d=1}^{M} N_d} \right) \tag{15}$$

Where $P(t_n|s_d)$ is the probability of having tag t_n given the d-th. service.

C. Results and Discussion

The choice of the number of topics corresponding to the original dataset has an impact on the interpretability of the results. In LocLDA and CorrLDA model the number of topics must be decided before training phase. There are several methods to choose the number of topics that lead to best general performance [26]. We evaluated the performance of our system using $AveragePrecision$ for increasing numbers of topics and the results peak at K = 70 (where K is the number of topics) before the performance starts to decrease. These evaluation results are shown in Figures 4 and 5. As observed from these figures, the better performance is obtained for the approach when the extracted and original tags are used to learne our models. We also evaluated the performance of our system by computing the perplexity of LocLDA and CorrLDA according to the three strategies discribed previousely. Figures 6 and 7 show the perplexity of the dataset test for each model by varying the number of topics (lower numbers are better). The results show that LocLDA and CorrLDA models achieve best performance when we mix the original tags and extracted tags.

As the manual creation of ground truth costs a lot of work, we use the 10% service of the dataset test and we generate top-10 tags using the probabilistic method based on CTM (Section III-A). The generated tags are considered as the true labels to evaluate the performance of our Web services tags ranking system. In addition, for each service in the dataset test, each of its tags is labeled as one of the five levels $relevance(s) \in \{1, 2, 3, 4, 5\}$ where 5 denotes *Most Relevant*, 4 denotes *Relevant*, 3 denotes *Partially Relevant*, 2 denotes *Weakly Relevant*, and 1 denotes *Irrelevant*.

The averaged $Precision@n$ and $NDCG_n$ were measured for up to the first ten generated tags from the complete list of results. These evaluation results are respectively shown in Figures 8 and 9. The results show that our approach performs better than the method based on CorrLDA model.

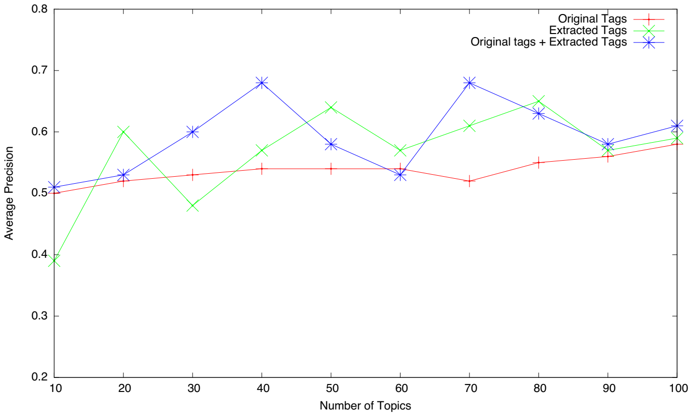


Fig. 4. Comparaison of average $Precision$ values over all dataset test for CorrLDA.

V. WS-Portal; An Enriched Web Services Search Engine

In this section, we describe some functionalities for our web services search engine where we incorporate our research works to facilitate web service discovery task. Our WS-Portal[4] contains 7063 providers, 115 sub-classes of category and 22236 web services crawled from the Internet [6]. In WS-Portal, severals technologies, i.e., web services clustering, tags recommendation, services rating and monitoring are employed

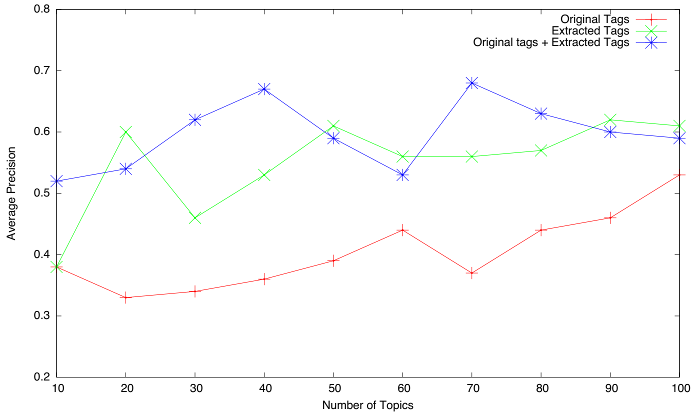


Fig. 5. Comparaison of average $Precision$ values over all dataset test for LocLDA.

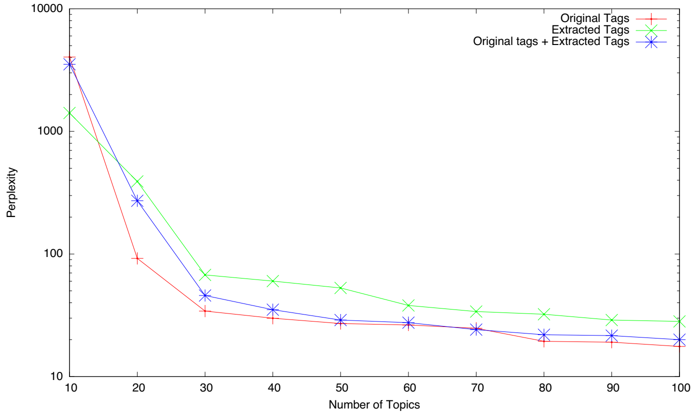


Fig. 6. Perplexity values obtained for learned CorrLDA model.

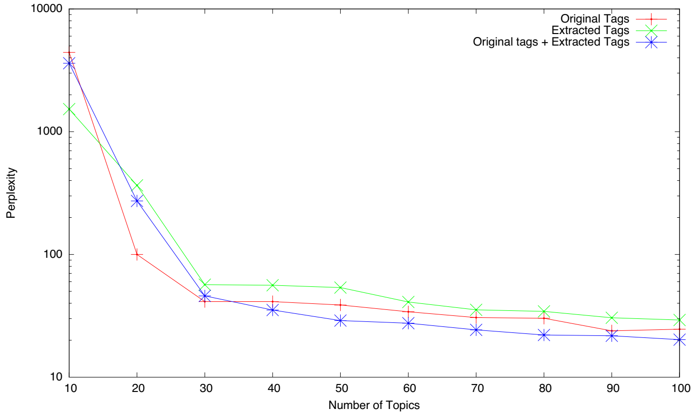


Fig. 7. Perplexity values obtained for learned LocLDA model.

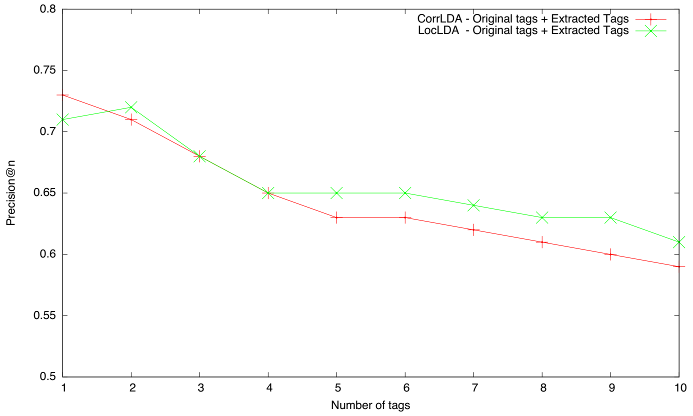


Fig. 8. Comparaison of average $Precision@n$ values for CorrLDA and LocLDA over the third dataset test (Original Tags and Extracted Tags).

to improve the effectiveness of web services discovery. Specifically, probabilistic topics models are utilized for clustering, services/topics and tags recommendation [3], [4], [5]. We use probabilistic topic models to extract topic from semantic service descriptions and search for services in a topics space where heterogeneous service descriptions are all represented as a probability distribution over topics.

A. Service Clustering

By organizing service descriptions into clusters, services become easier and therefore faster to discover and recommend. Web services are described as a distribution of topics [4]. A distribution over topics for a given service s is used to determine which topic best describes the service s. K clusters are created where K is the number of generated topics.

B. Service Discovery

Service Discovery and Selection aim to find web services with user required functionalities. A user query represented by a set of words is represented as a distribution over topics [3], [4], [5]. The service discovery is based on computing the similarity between retrieved topic's services and a user's query. We use the topics browsing technique as another method search to discover the web services that match with users requirements. Users can select the related topic to the their query and our system gives automatically the topic's services that match with user's query.

C. Tags recommendation

We use the automatic tagging technique proposed in this paper to recommend automatically the tags for all published services in our repository.

D. Availability and performance monitoring

WS-Portal monitor all registered services. In addition, after registering a service in our service registry its availability will be monitored automatically. Our system measures the availability by calling the service endpoints periodically.

E. Services rating and comments posting

Our system allows users to rate and post comments to enriche the service descriptions.

F. Dynamic service invocation

Our system allows users to invoke the selected service using the html form generated automatically from the associated WSDL document for each service operations.

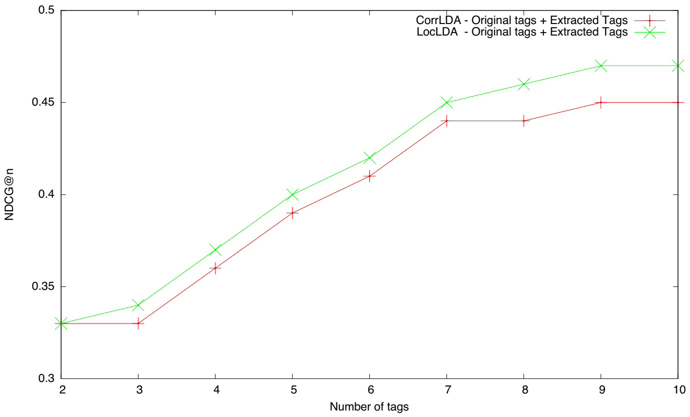


Fig. 9. Comparaison of average $NDCG@n$ values for CorrLDA and LocLDA over the third dataset test (Original Tags and Extracted Tags).

G. User Interface

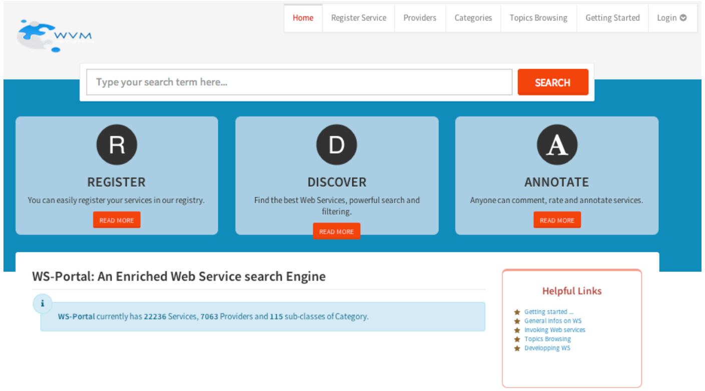
Our web services search engine is available online[4] and consumers can use it to discover, register or annotate web services. Figure 10 shows the site home page of our Web Services Search Engine. When users submit the search form, our system gives a list of services that match with user's query and each search result entity show a breif service description:

1) Web service name,
2) Service description,
3) Tags given by users,
4) Service category,
5) Service provider,
6) Average rating score given by users,
7) Service availability.

In addition our system select automatically a top five related topics to the user's query. When users select a disered service, WS-Portal gives more details for selected service such as service name, wsdl url, service documentation, provider, categories, country, availability, rating score, user's tags, recommended tags and WSDL cache. Our system gives also more details for service monitoring (availabilty and response time values for each service endpoints). In addition, users can rate, annotate the selected service and post comments. Finally, users can invoke the selected service using the html form generated automatically from WSDL document for each service operations. Our system gives also two others important informations such as similar services and the related topics to the selected service. Indeed, we use the extracted topics from services descriptions to calculate the similarity between the selected service and others web services in our repository. For this, we compute the similarity score, using some probability metrics such as *Cosine Similarity* and *Symmetric KL Divergence* [5], between the vectors containing the service's distribution over topics. Finally, similar services are ranked in order of their similarity score to the selected service. Thus, we obtain automatically an efficient ranking of the services retrieved.

VI. Conclusion

In this paper, we propose a novel approach based on probabilistic topic model to tag web services automatically.



WS-Portal: An Enriched Web Service search Engine

WS-Portal currently has 22236 Services, 7063 Providers and 115 sub-classes of Category.

Fig. 10. Site Home Page of Our Web Services Search Engine.

Three tags recommendation strategies are also developed to improve the system performance. Our system performs better when we mix the original tags and extracted tags from WSDL documents. A series experiments prove that our method is very effective. The comparisons of Precision@n, Normalised Discounted Cumulative Gain (NDCGn) values for our approach indicate that the method presented in this paper outperforms the method based on the CorrLDA in terms of ranking and quality of generated tags. We have presented also in this paper the Web Services Search engine developed to facilitate the service discovery process. In the future, we will focus our research on how to automatically tag RESTful services.

References

[1] Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services - Concepts, Architectures and Applications. Springer Verlag, Berlin Heidelberg, 2004.

[2] Azmeh, Z.; Falleri, J.-R.; Huchard, M. and Tibermacine, C.: Automatic Web Service Tagging Using Machine Learning and WordNet Synsets, in International Conference on Web Information Systems and Technologies (WEBIST 2010).

[3] Aznag, M., Quafafou, M. and Jarir, Z.: Correlated Topic Model for Web Services Ranking. In International Journal of Advanced Computer Science and Applications (IJACSA), vol. 4, no. 6, pp. 283–291, July 2013.

[4] Aznag, M., Quafafou, M., Rochd, El M., and Jarir, Z.: Probabilistic Topic Models for Web Services Clustering and Discovery. In the European Conference on Service-Oriented and Cloud Computing (ESOCC'2013), Springer LNCS 8135, pages 19-33, 11 September 2013.

[5] Aznag, M., Quafafou, M. and Jarir, Z.: Leveraging Formal Concept Analysis with Topic Correlation for Service Clustering and Discovery. In 21th IEEE International Conference on Web Services (ICWS 2014). Alaska, USA.

[6] Aznag, M., Quafafou, M. and Jarir, Z.: WS-Portal: An Enriched Web Services Search Engine. In 12th International Conference on Service Oriented Computing (ICSOC 2014), Paris, France.

[7] Blei, D., Ng, A. Y. and Jordan, M. I.: Latent dirichlet allocation. Journal of Machine Learning Research, vol. 3:993-1022, 2003.

[8] Blei, D., and Jordan, M.: Modeling annotated data. Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval, page 127-134. ACM Press, (August 2003)

[9] Chen, L.; Wang, Y.; Yu, Q.; Zheng, Z. and Wu, J.: WT-LDA: User Tagging Augmented LDA for Web Service Clustering., in 12th International Conference on Service Oriented Computing (ICSOC'2013), Springer LNCS 8274, pages 162-176.

[10] Chen, L., Wu, J., Zheng, Z., Lyu, M. R., Wu, Z.: Modeling and Exploiting Tag Relevance for Web Service Mining. in Knowledge and Information Systems Vol. 39, No. 1, pp 153-173. April 2014.

[11] Dong, X., Halevy, A., Madhavan, J., Nemes, E., Zhang, J.: Similarity Search for Web Services. In VLDB Conference, Toronto, Canada, pp. 372-383, 2004.

[12] Elgazzar, K., Hassan A., Martin, P.: Clustering WSDL Documents to Bootstrap the Discovery of Web Services. In IEEE International Conference on Web Services (ICWS'2010), pp. 147-154.

[13] Fang, L.; Wang, L.; Li, M.; Zhao, J.; Zou, Y. and Shao, L.: Towards Automatic Tagging for Web Services., in IEEE International Conference on Web Services (ICWS 2012).

[14] Gawinecki, M.; Cabri, G.; Paprzycki, M. and Ganzha, M., WSColab: Structured Collaborative Tagging for Web Service Matchmaking., in International Conference on Web Information Systems and Technologies (WEBIST 2010), pages 70-77.

[15] Hofmann, T.: Collaborative filtering via Gaussian probabilistic latent semantic analysis. In Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 259-266. ACM Press, 2003.

[16] Iwata, T., Yamada, T., and Ueda, N.: Probabilistic latent semantic visualization: topic model for visualizing documents. In KDD'2008: Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 363-371. ACM, 2008.

[17] Jordan, M.I., Ghahramani, Z., Jaakkola, T.S. and Saul, L.K: An introduction to variational methods for graphical models. *In Machine Learning*, 37:183–233, 1999.

[18] Kokash, N.: A Comparison of Web Service Interface Similarity Measures. Frontiers in Artificial Intelligence and Applications, Vol. 142, pp.220-231, 2006.

[19] Liu, Wei., Wong, W.: Web service clustering using text mining techniques. In International Journal of Agent-Oriented Software Engineering (IJAOSE'2009), Vol. 3, No. 1, pp. 6-26.

[20] Meyer, H. and Weske, M.: Light-Weight Semantic Service Annotations Through Tagging., in International Conference on Service Oriented Computing (ICSOC'2006), Springer LNCS 4294, pages. 465-470.

[21] Neal, R.M. and Hinton, G.E. A view of the EM algorithm that justifies incremental, sparse, and other variants. *In M.I. Jordan, editor, Learning in Graphical Models, pages 355368. Kluwer*, 1998.

[22] Porter, M. F.: An Algorithm for Suffix Stripping, In: Program 1980, Vol. 14, No. 3, pp. 130-137.

[23] Rochd, E. M., Quafafou, M.; Aznag, M.: Encoding Local Correspondence in Topic Models. In IEEE 25th IEEE International Conference on Tools with Artificial Intelligence (ICTAI), pp.602,609, Washington, 4-6 Nov 2013.

[24] Rosen-Zvi, M., Griffiths, T., Steyvers, M., Smyth, P.: The author-topic model for authors and documents. In 20th Conference on Uncertainty in Artificial Intelligence. pp. 487494 (2004)

[25] Salton, G.: Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA (1989).

[26] Steyvers, M. and Griffiths, T.: Probabilistic topic models. In Latent Semantic Analysis: A Road to Meaning, T. Landauer, D. Mcnamara, S. Dennis, and W. Kintsch, Eds. Laurence Erlbaum, 2007.

[27] W3C (2004). Web services architecture. Technical report, W3C Working Group Note 11 February 2004.

[28] Zheng, Z., Ma, H., Lyu, M.R., King, I.: QoS-aware Web service recommendation by collaborative filtering. IEEE Transactions on Service Computing 4(2), 140-152 (2011)