Semantic Pattern Mining Based Web Service Recommendation

Hafida Naïm, Mustapha Aznag, Nicolas Durand, and Mohamed Quafafou

Aix-Marseille University, CNRS, LSIS UMR 7296, 13397, Marseille, France.
hafida.naim@etu.univ-amu.fr
{mustapha.aznag, nicolas.durand, mohamed.quafafou}@univ-amu.fr

Abstract. This paper deals with the problem of web service recommendation. We propose a new content-based recommendation system. Its originality comes from the combination of probabilistic topic models and pattern mining to capture the maximal common semantic of sets of services. We define the notion of semantic patterns which are maximal frequent itemsets of topics. In the off-line process, the computation of these patterns is performed by using frequent concept lattices in order to find also the sets of services associated to the semantic patterns. These sets of services are then used to recommend services in the on-line process. We compare the results of the proposed system in terms of precision and normalized discounted cumulative gain with Apache Lucene and SAWSDL-MX2 Matchmaker on real-world data. Our proposition outperforms these two systems.

Keywords: Web services, Recommendation, Topic models, Formal concept analysis, Concept lattice, Maximal frequent itemsets.

1 Introduction

Web services[1] are defined as software systems designed to support interoperable machine-to-machine interaction over a network. They are "loosely coupled, reusable software components that semantically encapsulate discrete functionality and are distributed and programmatically accessible over standard Internet protocols". Web services are self contained, modular business applications that have open, internet-oriented and standards based interfaces. The explosion of web services with identical or similar functionalities over the internet has become a problem for the users. How can they find the best services that match their requirements from a large number of web services which have the requested functionality? Recommendation systems and selection techniques can be used to overcome this problem and assist users by recommending relevant web services from a large number of available web services [26].

Recent research efforts on web service recommendation focus on two approaches: collaborative filtering and content-based recommendation. Collaborative filtering approaches [26, 29] are used in almost all recommendation systems.

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

They find relevant services for the current user by collecting information from other similar users. For example, a list of services that many users like, can be used as recommendations for other users that share a large overlap of services with this list. Content-based approaches [5, 7] recommend web services on the basis of the similarity between the user request and the web service description (e.g., service functionalities). If the similarity between the user request and a service is high, this service is then recommended to the user.

In this paper, we propose a new content-based recommendation system. Its originality comes from the combination of probabilistic topic models and pattern mining to capture the maximal common semantic of sets of services. To the best of our knowledge, this is the first time that such approach combining the two domains is proposed. The core of the system is to identify the services which are very semantically linked. For this purpose, we defined the notion of semantic patterns. These latter correspond to maximal frequent itemsets of topics. Topics (or latent factors) correspond to a family of generative probabilistic models based on the assumption that documents (i.e., service descriptions) are generated by a mixture of topics where topics are probability distributions on words [23]. Topic models are used as efficient dimension reduction techniques which are able to capture semantic relationships between word-topic and topic-service [4]. The maximal frequent itemset discovery computes the maximal sets of items (i.e., topics), with respect to set inclusion, that appear together in at least a certain number of transactions (i.e., services) recorded in a database [12]. The semantic patterns allow to group together the services which are similar. Indeed, to each semantic pattern, the services containing this pattern can be associated. The services of a semantic pattern are very interesting: they are semantically linked and maximal. In order to compute semantic patterns and the corresponding sets of services, we used frequent concept lattices [27]. These sets of services are then stored in a special structure, called MFI-tree [11], in order to save space and perform quick searches by the recommendation engine. From a specified service, the recommendation engine uses this tree to find semantically similar services. The obtained services are then ranked and recommended to the user. For evaluation purposes, we conducted experiments on real-world data, and evaluated the quality of the recommended services. We also compared our system with two existing approaches: *Apache Lucene* and *SAWSDL-MX2 Matchmaker*.

The remainder of this paper is organized as follows. Section 2 provides an overview of related work. In Section 3 we describe in detail our service recommendation system. The experiments and the results are presented in Section 4. Finally, the conclusion and future work can be found in Section 5.

2 Related Work

Recommendation systems are assimilated to information filtering systems because the ideas and the methods are very close. We focus on two main types of filtering: content-based filtering and collaborative filtering. The interested reader can refer to [21] for further information about recommendation systems.

There is a lot of works on recommendation systems especially in the case of web navigation. So, we present some works in this context before considering the context of web services. Patterns are particularly used for collaborative filtering. These systems are based, for instance, on frequent itemsets, maximal frequent itemsets, clustering, formal concept analysis (i.e., concept lattices) or markov model [24]. The semantic aspects can be introduced in content-based approaches by using topic models or ontologies. In [20, 25], the authors have computed topic models. The probabilistic topic model is Latent Dirichlet Allocation (LDA). They do not use patterns. Let us note that we do not use LDA but Correlated Topic Model (see Section 3.1). A notion of semantic patterns has been proposed in [14] but the definition does not correspond to ours. They do not consider topics. A semantic pattern is a path that connects a source type to a target type through pairs property-type. Our definition is: semantic patterns are maximal frequent itemsets of topics.

Let us consider the context of web services. Generally, every web service has a WSDL (Web Service Description Language) document that contains the description of the service. To enrich web service descriptions, several Semantic Web methods and tools are developed, for instance, the authors of [22] use an ontology to annotate the elements in web services. Nevertheless, the creation and maintenance of ontologies may be difficult and involve a huge amount of human effort [1]. The content-based approaches and/or the non-logic-based semantic approaches [7, 13, 17, 18] aim to reduce the complexity of the discovery process by analysing the frequency of occurrence of some concepts and determine semantics which are implicit in service descriptions. These approaches generally use techniques such as information retrieval, data mining and linguistic analysis [17]. As the context of web navigation, the collaborative filtering approaches are widely used in web service recommendation systems [26, 28, 29]. In [29], the authors propose a collaborative filtering based approach for making personalized quality of service value prediction for the service users. In another context, Mehta et al. [16], propose an architecture for recommendation-based service mediation in which they take into account two more dimensions of service description: quality and usage pattern. The usage pattern permits to find applications with a similar usage pattern to the application making the request and then returns a recommendation list containing the services used by such applications.

As we can see, recommendation systems can use topic models or ontologies for considering semantics. Patterns are used especially for collaborative filtering and for capturing usages. The maximal frequent itemsets are not considered. We propose a content-based recommendation system leveraging probabilistic topic models and pattern mining (more precisely, maximal frequent itemset mining).

3 Web Service Recommendation System

In this section, we first give an overview of the proposed system. We then describe more in detail the different steps of our approach.

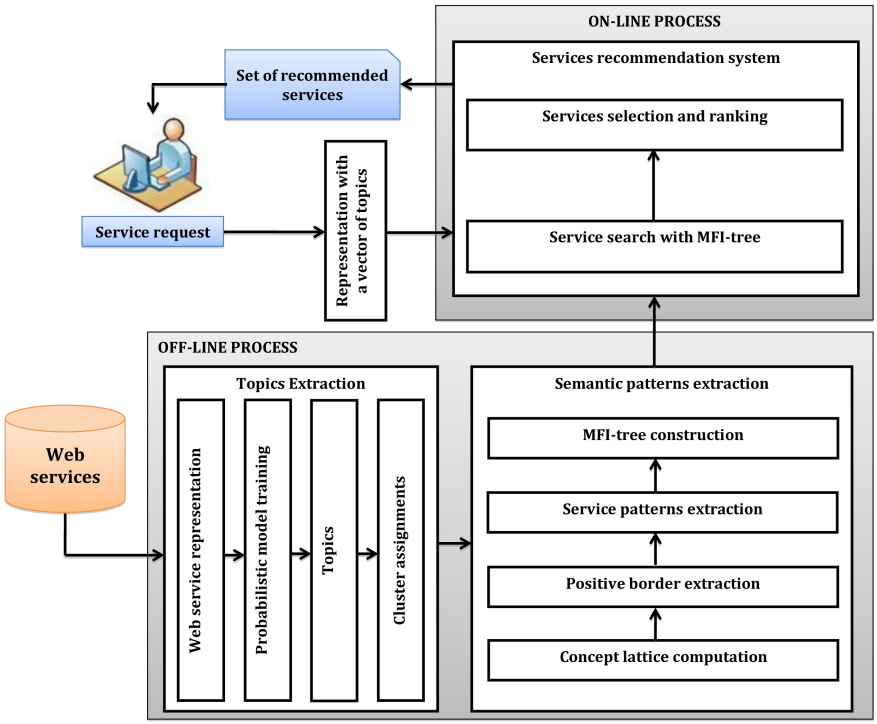

Fig. 1. Overview of the proposed recommendation system.

The proposed system relies on the notion of topics and semantic patterns. Topic models are used to capture semantic relationships between word-topic and topic-service. Semantic patterns capture the maximal common semantic of sets of services. The services corresponding to semantic patterns are used by the system. Let us note that this work extends our previous works on probabilistic web services clustering and discovery based on probabilistic topic models [2, 4].

Figure 1 shows the overview of our system with the different steps involved. As shown in this figure, we can distinguish two kinds of process: online process and offline process. The different steps of the offline process are listed as follows: (1) Topics extraction, (2) Semantic patterns extraction. Once all these tasks are done, we can easily recommend web services from a service selected by the user in the list of services returned by a discovery system. We note that this is the only task of the online process.

3.1 Topics extraction and cluster assignments

Topics (or latent factors) are a concept introduced by Probabilistic Topic Models [6]. They are a family of generative probabilistic models based on the assumption that documents are generated by a mixture of topics where topics are probability distributions on words. Topic models are used, in our context,

as efficient dimension reduction techniques, which are able to capture semantic relationships between word-topic and topic-service interpreted in terms of probability distributions. In [2, 4], we investigated the use of three probabilistic topic models PLSA, LDA and CTM [6] to extract topics from semantically enriched service descriptions and propose a probabilistic method for web services clustering and discovery. 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. In this paper, we use the ***Correlated Topic Model*** (CTM) [6] to extract latent factors from web service descriptions.

After the CTM model is trained, the distribution of textual concepts for each topic is known and all the services in the dataset can be described as a distribution of topics (i.e. a vector $\overline{s} = \{z_1, z_2, ..., z_K\}$ where each dimension z_k reflects the probability of that service description being generated by sampling from topic k. Let $\theta^{(s)}$ refer to the multinomial distribution over topics in the service description s and $\phi^{(j)}$ refer to the multinomial distribution over concepts for the topic z_j. We create K clusters where K is the number of generated topics (i.e. a cluster for each topic). The distribution over topics $\theta^{(s)}$ for service s is used to determine which topic best describes the service s. More precisely, if a probability distribution $\theta^{(s)}$ over a specific z_j when given a web service s is high, then the service s can be affected to the cluster C_j. If a service s has more than one topic, the service will be assigned to each of the clusters corresponding to these topics [3]. To simplify, we use the *multiple topics assignment strategy* to assign a set of topics for each service by selecting a *topK* topics. Thus, a service could be assigned to multiple clusters (e.g., the three best fitting clusters). This will increase the scope of each search. Multiple cluster assignments achieve higher recommendation accuracy. However, it comes at the cost of increased number of comparisons and computations (see Section 4).

3.2 Semantic pattern extraction

In order to define the notion of semantic patterns, we need to introduce some definitions. A data mining context is denoted by $\mathcal{D} = (\mathcal{T}, \mathcal{I}, \mathcal{R})$ where $\mathcal{T}$ is a set of transactions (i.e., web services), $\mathcal{I}$ is a set of items (i.e., topics), and $\mathcal{R} \subseteq \mathcal{T} \times \mathcal{I}$ is a binary relation between transactions and items. Each couple $(t, i) \in \mathcal{R}$ denotes the fact that the transaction t is related to the item i (e.g., t contains i). A transactional database is a finite and nonempty multi-set of transactions. Table 1 provides an example of such database consisting of 6 transactions (each one identified by its "Id") and 8 items (denoted $A \dots H$). In our context, services are transactions and topics are items. For each service, we assign the best topics (see Section 3.1). This assignment forms the binary relation $\mathcal{R}$.

An *itemset* is a subset of $\mathcal{I}$ (note that we use a string notation for sets, e.g., AB for $\{A, B\}$). An itemset is sorted in lexicographic order and is also called ***pattern***. A transaction t supports an itemset X iff $\forall i \in X, (t, i) \in \mathcal{R}$. An itemset X is *frequent* if the number of transactions which support it, is greater than (or is equal to) a minimum threshold value, noted *minsup*. The set of all-frequent

Table 1. Example of transactional database.

Id	Items					
1	A		C		E	G	
2		B	C		E	G	
3	A		C		E		H
4	A			D	F		H
5		B	C		F		H
6		B	C		E	F	H

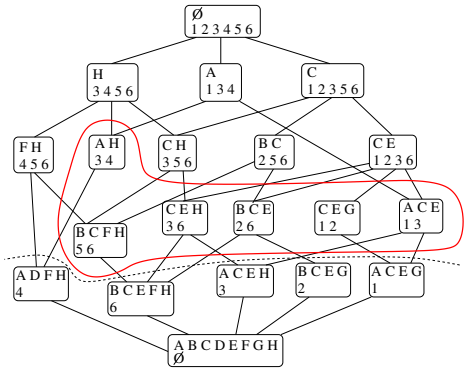

Fig. 2. Example of concept lattice (Bd^+ is encircled for $minsup$=2).

itemsets is $S = \{X \subseteq \mathcal{I},\ |\{t \in \mathcal{T},\ \forall i \in X \quad (t,i) \in \mathcal{R}\}| \quad \geq minsup\}$. The set of all ***maximal frequent itemsets*** (MFI), w.r.t. set inclusion, in $\mathcal{D}$ is the ***positive border*** of S, noted $Bd^+(S)$, and is equal to $\{X \in S \mid \forall Y \supset X, Y \notin S\}$ [15]. Let us take the example of Table 1, if $minsup$=2 then the itemset H is frequent because 4 transactions support it (3, 4, 5 and 6). BG is not frequent because only 2 supports it. CE is frequent but not maximal because CEH is also frequent. The set of MFIs is the positive border $Bd^+(S)$ and is equal to $\{AH, ACE, BCE, CEG, CEH, BCFH\}$.

A ***semantic pattern*** is a maximal frequent itemset of topics. To each semantic pattern, the transactions (i.e., services) containing this pattern can be associated. The services of a semantic pattern are very interesting: they are semantically linked and maximal. Thus, the proposed system uses these services. The minimum support threshold, $minsup$, allows to fix the minimum number of services for each semantic pattern. In order to extract the semantic patterns and their associated services, we compute the frequent concept lattice. Then, the set of services corresponding to each semantic pattern is selected and stored in a special structure called MFI-tree.

Concept lattice computation and positive border extraction Given $\mathcal{D}$, there is a unique ordered set which describes the inherent lattice structure defining natural groupings and relationships among the transactions and their related items. This structure is known as a concept lattice or Galois lattice [10]. Each element of the lattice is a couple (I, T) composed of a set of items (i.e., topics, the *intent*) and a set of transactions (i.e., services, the *extent*). Each couple (called ***formal concept***) must be a complete couple with respect to $\mathcal{R}$, which means that the following mappings (noted f and g) hold. For $T \subseteq \mathcal{T}$ and $I \subseteq \mathcal{I}$, we have: (1) $f(T) = \{i \in \mathcal{I} | \forall t \in T, (t,i) \in \mathcal{R}\}$ and (2) $g(I) = \{t \in \mathcal{T} | \forall i \in I, (t,i) \in \mathcal{R}\}$. $f(T)$ returns items common to all transactions $t \in T$, while $g(I)$ returns transactions that have at least all items $i \in I$.

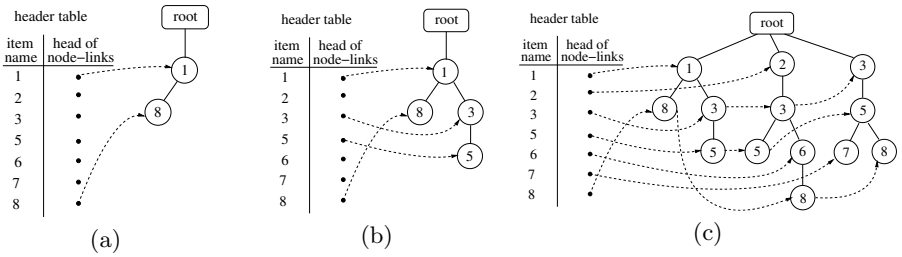

Fig. 3. MFI-tree construction.

The idea of maximally extending the sets is formalized by the mathematical notion of *closure* in ordered sets. The operators $h_1 = f \circ g$ and $h_2 = g \circ f$ are the Galois closure operators. Let X be an itemset, if $h_1(X)$=X, then X is a *closed itemset*. A formal concept is composed of a closed itemset and of the set of transactions containing this closed itemset. The ***frequent concept lattice*** is formed using the formal concepts that have at least *minsup* transactions in their extent. The "bottom" concept (i.e., ($\mathcal{I}$, $\emptyset$)) is kept. Due to the fact that the intents of the frequent formal concepts form the set of all-frequent closed itemsets [19] and that the set of all-maximal frequent itemsets is a subset of frequent closed itemsets, we can easily find $Bd^+(S)$ (i.e., the set of semantic patterns) from the frequent concept lattice. The positive border corresponds to the frequent formal concepts just above the bottom. Figure 2 presents the concept lattice obtained using the example of Table 1. The bottom is (A B C D E F G H, $\emptyset$). With *minsup*=2, the frequent formal concepts are above the dashed line. The formal concepts corresponding to the $Bd^+(S)$ are encircled. So, the semantic patterns are $\{AH, \ldots, BCFH\}$ and the corresponding sets of services are $\{\{3,4\}\{1,3\}\{1,2\}\{2,6\}\{3,6\}\{5,6\}\}$. Let us remark that the concepts of $Bd^+(S)$ can have more than *minsup* transactions in their extent (see Section 4.3).

Service pattern extraction and MFI-tree construction The result of the previous step is the set of formal concepts corresponding to the $Bd^+(S)$ (i.e., the set of semantic patterns). The proposed system selects the extents of these formal concepts to form the sets of services which will be used by the online recommendation engine. These sets of services are considered as patterns. To facilitate the recommendation, we store these patterns of services in a variant of FP-tree (Frequent Pattern tree) called MFI-tree (Maximal Frequent Itemsets tree) [11]. This allows a space saving and a quick search of the patterns containing a given service by using indexes. Every branch of the tree represents a pattern. Compression is acheived by building the tree in such way that overlapping patterns share prefixes of the corresponding branch. The tree has a root labelled with "root". Children of the root are item prefix subtrees. Each node in the subtree has four fields: item-name, children-list, parent-link and node-link. All nodes with same item-name are linked together. The node-link points to

the next node with same item-name. A header table is constructed for items in the MFI-tree. Each entry in the header table consists of two fields, item-name and head of a node-link. The node-link points to the first node with the same item-name in the MFI-tree. Let us take a new example (more complete than the first one) where we have extracted the semantic patterns and then found these patterns of services: $\{\{1,8\}\{1,3,5\}\{2,3,5\}\{3,5,7\}\{3,5,8\}\{2,3,6,8\}\}$. Figure 3 illustrates the construction of the tree. We get the first pattern $\{1,8\}$. It is inserted into the tree directly (see Figure 3 (a)). We then insert $\{1,3,5\}$ into the tree (see Figure 3 (b)). Figure 3 (c) presents the complete tree.

3.3 Web service recommendation task

From a service s, the proposed system find the services present with s in the patterns of services computed in the offline process. These services are ranked and recommended to the user. Algorithm 1 present the search of recommended services from a service s by using the MFI-tree constructed in the previous step. It returns the items (i.e., services) present in the patterns containing s. The idea of the algorithm is to use the header table of the tree to access directly to the different patterns containing the item s. For each node N corresponding to s (Step 2), we need to find the common prefix (PX) of the patterns (Steps 3 to 8). It corresponds to go up to the root node via the parent links. Then we find all the possible ends of the patterns (i.e., the suffixes SX) (Step 10). The items of the prefix and of the suffixes are merged (Steps 11 and 12) and will be returned at the end of the algorithm. Let us take an example: the service 5 and the tree of Figure 3 (c). For the first node corresponding to 5, PX={1, 3} and SX={}, we have R={1, 3}. For the second node, PX={2, 3} and SX={}, so we have R={1, 2, 3}. For the last node, PX={3} and SX={{7}, {8}}. The services R to recommend are {1, 2, 3, 7, 8}. Let us note that it is possible to recommend services from a set of services S by intersecting the set of recommended services obtained for each service $s \in S$.

Once the recommended services are discovered using Algorithm 1, these services are ranked in order of their similarity score to the service request. Thus, we obtain automatically an efficient ranking of the recommended services. In our approach, we use the proximity measure called *Multidimentional Angle* (also known as *Cosine Similarity*); a measure which uses the cosine of the angle between two vectors. We calculate the similarity between the service request and each recommended web service by computing the Cosine Similarity between a vector containing the service request distribution over topics q and a vector containing the recommended service's distribution of topics p. The multidimensional angle between a vector p and a vector q can be calculated using Equation 1 where t is the number of topics.

$$Cos(p,q) = \frac{p.q}{\parallel p \parallel . \parallel q \parallel} = \frac{\sum_{i=1}^{t} p_i q_i}{\sqrt{\sum_{i=1}^{t} p_i^2 \sum_{i=1}^{t} q_i^2}}. \quad (1)$$

Algorithm 1 MFI-tree based web service recommendation algorithm

Require:
- **s** : a service
- **T** : the MFI-tree containing the patterns of services

Ensure: **R**: the set of recommended services

```
1: N ← T.header-table[s]; // node N: head of node links for s
2: while N != null do
3:    Parent ← N.parent-link; // parent node of N
4:    PX ← ∅; // common prefix
5:    while Parent != null do
6:       PX ← PX ∪ {Parent.item-name};
7:       Parent ← Parent.parent-link;
8:    end while
9:    SX ← ∅; // set of patterns starting from N
10:   findSuffixes(N, ∅, SX); // find patterns starting from N
11:   merge(SX); // union of all the patterns contained in SX
12:   R ← R ∪ PX ∪ SX; // add services to recommend
13:   N ← N.node-link; // next node corresponding to s
14: end while
15: return R;
```

The multidimensional angle takes values in the interval [0, 1] where 0 indicates no similarity and 1 indicates identical vectors.

4 Evaluation

4.1 Web services corpus and data preprocessing

The experiments are performed out based on real-world web services obtained from the WSDL service retrieval test collection called *SAWSDL-TC3*[2]. The WSDL corpus consists of 1088 semantically annotated WSDL 1.0-based Web services which cover 9 different application domains. Each web service belongs to one out of nine service domains named as: Communication, Education, Economy, Food, Geography, Medical, Military, Travel and Simulation. The dataset contains 42 queries (i.e., requests). A service request is defined as a service that would perfectly match the request. Furthermore, a binary and graded relevance set for each query is provided which can be used in order to compute Information Retrieval (IR) metrics. The relevance sets for each query consists of a set of relevant services and each service s has a graded relevance value $relevance(s) \in \{1, 2, 3\}$ where "3" denotes *high relevance* to the query and "1" denotes a *low relevance*. Table 2 lists the number of services and requests from each domain.

To manage efficiently web service descriptions, we extract all features that describe a web service from the WSDL document. Before representing web services as a *TF-IDF* (Text Frequency and Inverse Document Frequency) vectors,

[2] `http://www.semwebcentral.org/projects/sawsdl-tc`

Table 2. Number of services and queries for each domain.

Domain	Services	Queries	Domain	Services	Queries
Communication	58	2	Medical	73	1
Economy	358	12	Military	40	1
Education	285	6	Simulation	16	3
Food	34	1	Travel	164	6
Geography	60	10			

we need some preprocessing. The objective of this preprocessing is to identify the textual concepts of services, which describe the semantics of their functionalities. There are commonly several steps: *Features extraction*, *Tokenization*, *Tag and stop words removal*, *Word stemming* and *Service Transaction Matrix construction* (see [2] for more details). 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 IR, 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 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*. In the service matrix, 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 equation: $w_{ij} = tf_{ij}.\log(\frac{n}{n_j})$ 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. The observed textual concepts are represented in a Service Transaction Matrix (STM).

4.2 Protocol and Evaluation Metrics

To compute topics, we use the STM as training data for our implementation of the CTM model (based on the Blei's implementation[3], which is a C implementation of CTM using Variational EM for Parameter Estimation and Inference).

We analyse the impacts of the parameters *minsup* (i.e., the minimum support threshold) and *assign* (i.e., number of topic assignments) on the quality of the recommendations. For some *minsup* values and for some *assign* values, we adopted the following protocol: For the offline part: (1) Computation of the semantic patterns (by using CHARM-L [27] to generate the frequent concept lattice), (2) Extraction of the patterns of services, (3) Construction of the MFI-tree. The steps to simulate the online part are: For each query present in the dataset: (4) Search the recommended services by using Algorithm 1, (5) Ranking

[3] `http://www.cs.princeton.edu/~blei/ctm-c/index.html`

of the list of recommended services, (6) Evaluation of the quality of the first n recommended services.

In order to compare our web service recommendation system (labelled *Topic-MFI*) to two existing systems, Step 4 is redone twice by replacing our system by a syntax-based approach powered by *Apache Lucene*[4] and a method from the *SAWSDL-MX2 Matchmaker*[5] hybrid semantic matchmaker for SAWSDL services, respectively.

In the test collection, we have the queries together with the correct/expected web services (see Section 4.1). Thus, we estimate how well is a recommendation method by discovering services corresponding to each query in the data. After that, we compare the returned list of services with the expected one. Finally, we evaluate the accuracy of the recommendation system by using standard measures used in IR. Generally, the top most relevant retrieved services are the main results which are selected and used by the user. Thus, we evaluated the quality of the first n recommended services by computing *Precision at n* (*Precision@n*) and *Normalized Discounted Cumulative Gain* ($NDCG_n$). These are standard evaluation techniques used in IR to measure the accuracy of a search and matchmaking mechanism.

In our context, $Precision@n$ is a measure of the precision of the service discovery system taking into account the first n retrieved services. Therefore, $Precision@n$ reflects the number of services which are relevant to the user query. The $Precision@n$ for a list of retrieved services is given by Equation 2 where the list of relevant services to a given query is defined in the collection.

$$Precision@n = \frac{|RelevantServices \cap RetrievedServices|}{|RetrievedServices|}. \tag{2}$$

$NDCG_n$ uses a graded relevance scale of each retrieved service from the result set to evaluate the gain, or usefulness, of a service based on its position in the result list. This measure is particularly useful in IR for evaluating ranking results. The $NDCG_n$ for n retrieved services is given by Equation 3 where DCG_n is the Discounted Cumulative Gain and $IDCG_n$ is the Ideal Discounted Cumulative Gain.

$$NDCG_n = \frac{DCG_n}{IDCG_n}, \qquad DCG_n = \sum_{i=1}^{n} \frac{2^{relevance(i)} - 1}{log_2(1+i)}. \tag{3}$$

The $IDCG_n$ is found by calculating the DCG_n of the first n returned services. n is the number of retrieved services and $relevance(s)$ is the graded relevance of the service in the ith position in the ranked list. The $NDCG_n$ values for all queries can be averaged to obtain a measure of the average performance of a ranking algorithm. $NDCG_n$ values vary from 0 to 1. $NDCG_n$ gives higher scores to systems which rank a search result list with higher relevance first and penalizes systems which return services with low relevance.

In addition to these metrics, we also compute some statistics (the number of computed patterns and the average size of a pattern) and we measure the query

[4] `http://lucene.apache.org/`

[5] `http://projects.semwebcentral.org/projects/sawsdl-mx`

response times. All experiments were performed on a personal computer with a Intel Core2Duo processor, 2.4GHz, and 6GB of RAM.

4.3 Results and Discussion

Figure 4 presents the comparaison of average Precision@n values over 42 queries obtained for our method with different values of *minsup* (1 to 6) and *assign* (2 to 10 topics assigned to each service). A low or a high value of *assign* does not give the best results. The worst precision is obtained with $assign = 7$. Our method gives the higher precision values with $assign = 4$ (for each *minsup* values). So, we investigated more precisely the system when the *assign* value is equal to 4.

Table 3. Number and size of the patterns obtained for assign-4 (according to *minsup*).

minsup	**# patterns of services**	**Avg. size of a pattern**
1	307	3.54
2	205	5.01
3	159	6.37
4	137	7.21
5	111	8.71
6	101	9.71

Table 3 shows the number of service patterns obtained and the average number of services in a pattern, for $assign = 4$ and *minsup* varying from 1 to 6. As we can expected, the more the *minsup* value is low, the more the number of patterns is high. The average size of a pattern is more interesting. For instance, if *minsup* is equal to 1, a pattern can contain only one service. Nevertheless, we can observe that the average number of services is higher than the *minsup* value. The services are often correlated. Our system is able to find these correlations and is not restricted to the *minsup* value.

Figure 5 (left) and (right) present the average $Precision@n$ and $NDCG@n$ values, respectively. These measures are obtained over all 42 queries for our method *Topic-MFI*, *ApacheLucene* and *SAWSDL-MX2 Matchmaker*. In both cases, the results show that *Topic-MFI* gives a higher average $Precision@n$ and $NDCG_n$ for all 42 queries. In fact, our method perform better than all methods. The results show that *ApacheLucene* and *SAWSDL-MX2* were unable to find some of the relevant web services that were not directly related to some of the requests through keywords or logic descriptions. This reflects that the retrieved services obtained by our method are specific to the user's query. *ApacheLucene* and *SAWSDL-MX2* have a low $NDCG_n$ because, as shown in the $Precision@n$ results, both approaches are unable to find some of the highly relevant services. The results obtained for our method reflect the accuracy of our recommendation system.

Table 4 presents the average query response times for *ApacheLucene*, *SAWSDL-MX2* and our method (*Topic-MFI*) for all 42 queries. As we can see, *Topic-MFI*

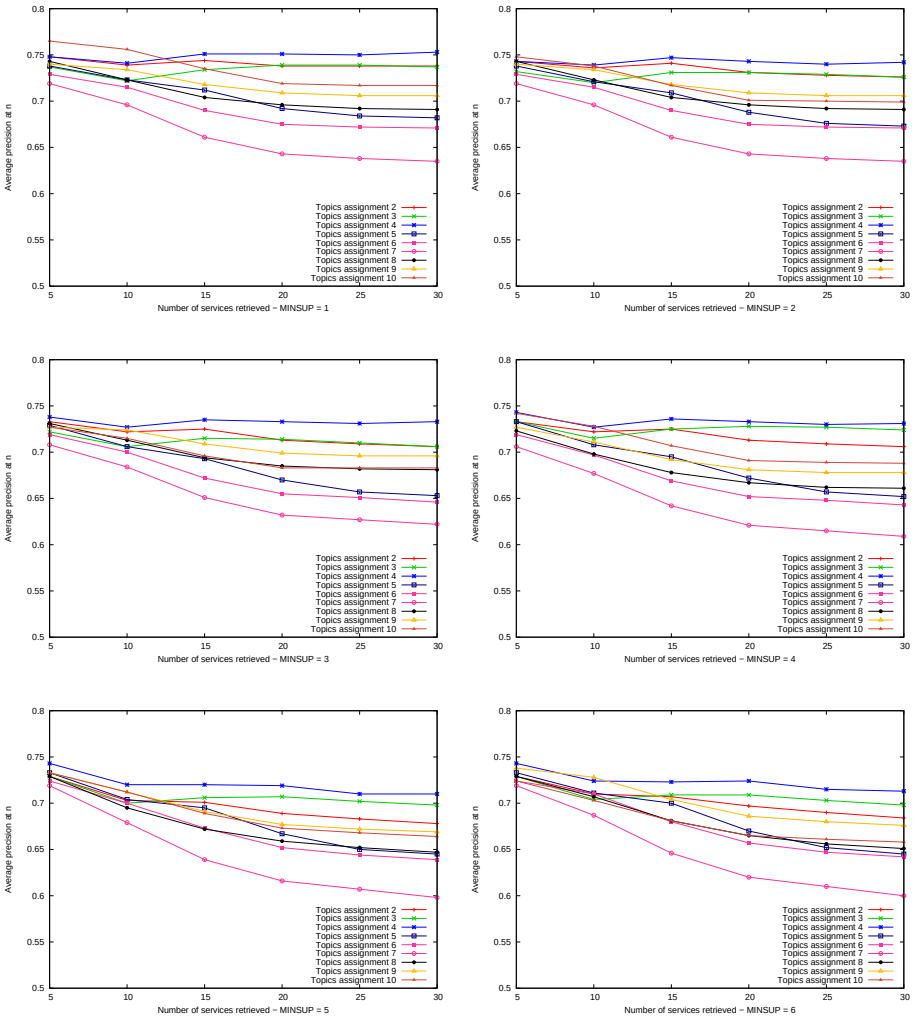

Fig. 4. Comparaison of average Precision@n values over 42 queries obtained for our method with different values of *minsup* and *assign* (# topics assigned to each service).

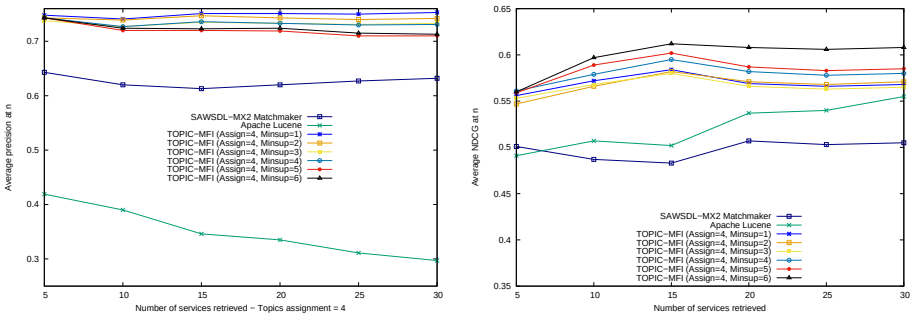

Fig. 5. (Left) Comparaison of average Precision@n values over 42 queries. (Right) Comparaison of average NDCGn values over 42 queries obtained for our method Topic-MFI and other baseline methods.

gives a faster query response time than the other search methods. Our recommendation system is efficient and not time-consuming.

Table 4. Average query response times.

Method	**Avg. query response time (ms)**
Topic-MFI	68
ApacheLucene	1163
SAWSDL-MX2	3045

5 Conclusion

We have introduced a new content-based recommendation system leveraging probabilistic topic models and pattern mining. Its originality comes from the combination of the two domains for capturing the maximal common semantic of sets of services. For this purpose, we defined the notion of semantic patterns which are the maximal frequent itemsets of topics. To compute these patterns and the corresponding sets of services, we used frequent concept lattices. In order to save space and perform quick searches among the computed sets of services, the system stores them in a special structure, called MFI-tree. The recommendation engine uses this tree to find services from a specified service. The obtained services are ranked and recommended to the user. The experimental results obtained on real-world web services show that our system outperforms *ApacheLucene* and *SAWSDL-MX2 Matchmaker*. In future work, we will use the approximation of frequent itemset border [8, 9] in order to extend our system and recommend supplementary services based on approximate semantic patterns.

References

1. Atkinson, C. ands Bostan, P., O., H., Stoll, D.: A Practical Approach to Web Service Discovery and Retrieval. In: ICWS. pp. 241–248 (2007)
2. Aznag, M., Quafafou, M., Jarir, Z.: Correlated topic model for web services ranking. IJACSA 4(6), 283–291 (2013)
3. Aznag, M., Quafafou, M., Jarir, Z.: Leveraging Formal Concept Analysis with Topic Correlation for Service Clustering and Discovery. In: ICWS (2014)
4. Aznag, M., Quafafou, M., Rochd, E.M., Jarir, Z.: Probabilistic Topic Models for Web Services Clustering and Discovery. In: ESOCC. LNCS, vol. 8135, pp. 19–33. Springer (2013)
5. Blake, M.B., Nowlan, M.F.: A Web Service Recommender System Using Enhancing Syntactical Matching. In: ICWS (2007)
6. Blei, D.M., Lafferty, J.D.: A Correlated Topic Model of Science. Annals of Applied Statistics pp. 17–35 (2007)
7. Cassar, G., Barnaghi, P., Moessner, K.: Probabilistic Matchmaking Methods for Automated Service Discovery. TSC 7(4) (2013)
8. Durand, N., Quafafou, M.: Approximation of Frequent Itemset Border by Computing Approximate Minimal Hypergraph Transversals. In: DaWaK. LNCS, vol. 8646, pp. 357–368. Springer (2014)
9. Durand, N., Quafafou, M.: Frequent Itemset Border Approximation by Dualization. T. Large-Scale Data- and Knowledge-Centered Systems 26, 32–60 (2016)
10. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical foundations. Springer (1999)
11. Grahne, G., Zhu, J.: Fast Algorithms for Frequent Itemset Mining Using FP-Trees. TKDE 17(10), 1347–1362 (2005)
12. Han, J., Cheng, H., Xin, D., Yan, X.: Frequent pattern mining: current status and future directions. Data Mining and Knowledge Discovery 15, 55–86 (2007)
13. Ma, J., Zhang, Y., He, J.: Efficiently Finding Web Services Using a Clustering Semantic Approach. In: CSSSIA. pp. 1–8 (2008)
14. Maccatrozzo, V., Ceolin, D., Aroyo, L., Groth, P.: A Semantic Pattern-Based Recommender, pp. 182–187. Springer International Publishing, Cham (2014)
15. Mannila, H., Toivonen, H.: Levelwise Search and Borders of Theories in Knowledge Discovery. Data Mining and Knowledge Discovery 1(3), 241–258 (1997)
16. Mehta, B., Niedere, C., Stewart, A., Muscogiuri, C., Neuhold, E.J.: An Architecture for Recommendation Based Service Mediation. In: ICSNW. LNCS, vol. 3226, pp. 250–262. Springer (2004)
17. Mohebbi, K., Ibrahim, S., Khezrian, M., Munusamy, K., Tabatabaei, S.G.H.: A Comparative Evaluation of Semantic Web Service Discovery Approaches. In: iiWAS. pp. 33–39 (2010)
18. Nayak, R., Lee, B.: Web service Discovery with Additional Semantics and Clustering. In: WI. pp. 555–558 (2007)
19. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Efficient Mining of Association Rules Using Closed Itemset Lattices. Information Systems 24(1), 25–46 (1999)
20. Qinjiao Mao, Q., Feng, B., Pan, S.: Modeling User Interests Using Topic Model. JATIT 48(1), 600–606 (2013)
21. Ricci, F., Rokach, L., Shapira, B., Kantor, P.B.: Recommender Systems Handbook. Springer, 1st edn. (2011)
22. Sivashanmugam, K., Verma, A., Miller, J.: Adding Semantics to Web services Standards. In: ICWS. pp. 395–401 (2003)

23. Steyvers, M., Griffiths, T.: Latent Semantic Analysis: A Road to Meaning, chap. Probabilistic topic models (2007)
24. Suguna, R., Sharmila, D.: An Efficient Web Recommendation System using Collaborative Filtering and Pattern Discovery Algorithms. IJCA 70(3), 37–44 (2013)
25. Xu, G., Zhang, Y., Yi, X.: Modelling User Behaviour for Web Recommendation Using LDA Model. In: WI-IAT. pp. 529–532 (2008)
26. Yao, L., Sheng, Q.Z., Ngu, A.H.H., Yu, J., Segev, A.: Unified Collaborative and Content-Based Web Service Recommendation. TSC 8(3), 453–466 (2015)
27. Zaki, M., Hsiao, C.J.: Efficient algorithms for mining closed itemsets and their lattice structure. TKDE 17(4), 462–478 (2005)
28. Zheng, Z., Ma, H., Lyu, M.R., King, I.: WSRec: A Collaborative Filtering Based Web Service Recommender System. In: ICWS. pp. 437–444 (2009)
29. Zheng, Z., Ma, H., Lyu, M.R., King, I.: QoS-Aware Web Service Recommendation by Collaborative Filtering. TSC 4(2), 140–152 (2011)