Leveraging Formal Concept Analysis with Topic Correlation for Service Clustering and Discovery

Mustapha AZNAG*, Mohamed QUAFAFOU* and Zahi JARIR**

* Aix-Marseille University, CNRS, LSIS UMR 7296, 13397 Marseille, France. {mustapha.aznag, mohamed.quafafou}@univ-amu.fr
** University of Cadi Ayyad, LISI Laboratory, FSSM, Morocco. jarir@uca.ma

Abstract—With a growing number of web services, discovering services that can match with a user’s query becomes a challenging task. It’s very tedious for a service consumer to select the appropriate one according to her/his needs. In this paper, we propose a non-logic-based matchmaking approach that uses the Correlated Topic Model (CTM) to extract topic from semantic service descriptions and model the correlation between the extracted topics. Based on the topic correlation, service descriptions can be grouped into hierarchical clusters. In our approach, we use the Formal Concept Analysis (FCA) formalism to organize the constructed hierarchical clusters into concept lattices according to their topics. Thus, service discovery may be achieved more easily using the concept lattice. In our approach, topic models are used 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 our experiment, we compared the accuracy of the our hierarchical clustering algorithm with that of a classical hierarchical agglomerative clustering. The comparisons of Precision@n and Normalised Discounted Cumulative Gain (NDCGn) values for our approach, Apache lucene and SAWSDL-MX2 Matchmaker indicate that the method based on CTM presented in this paper outperform all the others matchmakers in terms of ranking of the most relevant services.

Keywords—Web service, Data Representation, Discovery and ranking, Hierarchical Clustering, Machine Learning, Topic Models, Formal Concept Analysis.

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 services¹ [17] 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 [9]. 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 [14], [3]. With a growing number of web services, discovering services that can match with a user’s query becomes a challenging task. Automated service discovery is an important aspect as many high-level service oriented concepts such as service composition and service recommendation highly rely on the precision of an underlying automated service search engine. To enrich web service description, several Semantic Web methods and tools are developed, for instance, the authors of [15] use 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 [2]. The most of the semantic service discovery approaches are now based on semantic service description models [13]. These works are split into three main categories: logic-based approaches, non-logic based approaches, and hybrid approaches.

The key problem of the logic-based and hybrid approaches is the complexity of discovery process which makes it an intractable process over large service datasets [11]. The Non-logic-based semantic approaches [12], [10], [11], [7], 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 (IR), data mining and linguistic analysis [11].

In this paper, we propose a non-logic-based matchmaking approach that uses correlated topic model [6] to extract topic from semantic service descriptions and search for services in a topic space where heterogeneous service descriptions are all represented as a probability distribution over topics. Our proposed approach is built upon our previous work on representing service descriptions in terms of topics [4], [5]. Topics or latent factors are a concept introduced by Probabilistic Topic Models [16]. In [4], [5], we investigated the use of three probabilistic topic models PLSA, LDA and CTM [6] to extract latent factors from semantically enriched service descriptions. These latent factors 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

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

The Correlated Topic Model (CTM) has been developed to address the limitation of LDA [6]. In CTM, topic proportions exhibit correlation via the logistic normal distribution. In this paper, we utilized CTM to extract latent factors and the correlation between these topics to propose an efficient Web Service Hierarchical Clustering. Based on the hierarchical nodes (i.e. clustered groups) and the extracted topics, a set of matched services can be returned by comparing the similarity between the query and the related cluster, rather than computing the similarity between the query and each service in the dataset. In our approach, we use the Formal Concept Analysis (FCA) [3] formalism to organize the constructed hierarchical clusters into concept lattices according to their topics. Thus, service discovery may be achieved more easily using the concept lattice.

The rest of this paper is organized as follows. In Sections II and III we respectively describe in detail our Hierarchical clustering and Service Discovery Approach. Section IV describes the experimental evaluation. Finally, the conclusion and future work can be found in Section V.

II. TOPIC CORRELATION BASED HIERARCHICAL CLUSTERING

The number of web services created and published in a registry increases. Thus, searching services that can match with a user query becomes a challenging task. Comparing a user query to all services published in a service repository can be computationally expensive in large datasets. By organizing service descriptions into clusters, services become easier and therefore faster to discover and recommend [14]. Service Clustering aims to group together services which are similar to each other.

The work described in this section extends our previous work on probabilistic web service clustering based on correlated topic models [5]. The key idea of our approach is to cluster the services into a group of learned latent factors and re-organize the linear structure into a hierarchical structure. Thus, a hierarchical structure provides a view of the data at different levels of abstraction and are widely used in knowledge representation, resources organization or document indexing. Hierarchical clustering [18] is a clustering analysis method designed for building a hierarchy of clusters. The basic idea behind the hierarchical clustering is to merge consecutively similar clusters into one, until a root is reached. The rationale for this is that the dimensionality of the model is reduced as every web service can be described in terms of a small number of topics rather than a large number of textual concepts. With the maximum value of the computation used for the cluster for a service, we can categorize services into their corresponding group. Consequently, searching for a service inside a cluster can be performed by searching for matching clusters rather than matching the text describing the web service to a set of keywords extracted from the user query.

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. Let \( \theta(s) = P(z) \) refer to the multinomial distribution over topics in the service description \( s \) and \( \phi^{(j)} = P(c|z_j) \) 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 (i.e. \( P(s|z_j) \)), 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. (See Algorithm 1).

Algorithm 1 Cluster assignments

Requirements:
- \( S = \{s_1, \ldots, s_M\} \) web services set. (\( M \) number of services).
- \( K \) Number of Topics.
- \( \epsilon_1 \) Threshold for topics distribution over services.

Ensure: \( K \) Clusters of services.
1: Perform CTM on services set \( S = \{s_1, \ldots, s_M\} \).
2: for each topic \( z_k \in Z = \{z_1, \ldots, z_K\} \) do
3:   for each service \( s_i \in S = \{s_1, \ldots, s_M\} \) do
4:     Compute \( P(s_i|z_k) \)
5:     if \( P(s_i|z_k) > \epsilon_1 \) then
6:       Assign the service \( s_i \) to Cluster \( c_k \)
7:     end if
8:   end for
9: end for
10: return Set of \( K \) clusters.

As mentioned previously, the ability of the CTM to model the correlation between topics yields a better fit of a services collection. The covariance of the logistic normal model for topic proportions can be used to visualize the relationships among the topics. In particular, the covariance matrix can be used to form a topic graph, where the nodes represent individual topics, and neighboring nodes represent highly related topics [6]. We exploit the concept of neighborhood in complex graphs to build the hierarchy of clusters. Two nodes \( i \) and \( j \) are \( l \)th neighbors, or neighbors of order \( l = 1, 2, \ldots \) if and only if the shortest path between them has length \( l \). We denote this by \( (i,j) \in O(l) \). In our work we identified, for each \( l \), all \( O(l) \) neighbors of \( G \) and constructed a family of graphs. Each \( G_l \) is defined by the same set of nodes, while links are inserted between all pairs of vertices that are \( O(l) \) neighbors in \( G = G_1 \). The \( G_l \)'s were actually set up by a corresponding family \( M_l \) of adjacency matrices (AM) \( M_l \) [1], achieved by the systematic use of Boolean (B) operations among matrices (i.e. Table I).

After the CTM model is trained and all services are assigned to related clusters, we use the correlation between the extracted
Based on the constructed clusters (Algorithm 1) and the Correlation Matrix (i.e. $M_1$), we group the clusters in an agglomerative hierarchical clustering style. An abstraction of this mechanism is shown in Figure 1. So, the whole algorithm is divided into three phases:

- Construct a flat clustering using Algorithm 1.
- Construct a distance matrix for all topics (i.e. $DM$) using the distance values based on the one of the probability metrics for hierarchical clustering’s distance measure (See Section IV-C2). Each topic $z$ can be described as a distribution of services (i.e. a vector $z = \{s_1, s_2, ..., s_M\}$) where each dimension $s_k$ reflects the probability of that service $k$ being generated by sampling from topic $z$.
- For each level we calculate the higher order matrix $M_l$ and for all closest pair in $M_l$ we group the clusters in an agglomerative hierarchical clustering style. During the clustering process, a pair of closest clusters is merged for each iteration. The whole algorithm is presented as Algorithm 2.
- Finally, we define the root cluster node as a combination of top level clusters.

### Algorithm 2 Hierarchical clustering algorithm

**Require:**
- $M_1$ Correlation Matrix;
- $DM$ Distance Matrix
- $\epsilon$ Threshold for distance measure.

**Ensure:** Hierarchy of clusters.

1. $l = 1$
2. $M_l = M_1$
3. $HClustersMap = new \text{Map}()$
4. **repeat**
5. \quad $\text{NewClusterSet} = new \text{Set}()$
6. \quad $\text{Calculate the higher order matrix } M_l \text{ of level } l \text{ of neighborhood using Equation 2.}$
7. \quad **for** each topic $t_x$ find its closest $t_y$ ($M_l(t_x, t_y) = 1$) **do**
8. \quad \quad \quad if $(DM(t_x, t_y) \leq \epsilon)$ **then**
9. \quad \quad \quad \quad **for** each set $e \in HClustersMap.get(l - 1)$ **do**
10. \quad \quad \quad \quad \quad if set.contains($t_x$) || set.contains($t_y$) **then**
11. \quad \quad \quad \quad \quad \quad merge the clusters $C_x$, $C_y$ and set to a new cluster $C_{new}$.
12. \quad \quad \quad \quad \quad \quad NewClusterSet.add($C_{new}$);
13. \quad \quad \quad end if
14. \quad \quad \quad end for
15. \quad end if
16. end for
17. $HClustersMap.put(l, \text{NewClusterSet})$
18. **until** $M_l = {}$
19. **return** $HClustersMap$.

**III. Probabilistic Service Discovery and Ranking**

Service Discovery and Selection aim to find web services with user required functionalities. While Service Discovery
process assumes that services with similar functionalities should be discovered, Service Selection and Ranking aim to find proper services with the best user desired quality of services. Thus, Service Ranking aims to give a value of relevance to each service returned by the discovery process and proceeds to order the results in descending order starting from the most relevant ones.

![Abstract representation of the probabilistic service matchmaking and ranking](image)

Based on the clustered service groups (i.e. hierarchical nodes), a set of matched services can be returned by comparing the similarity between the query and the related cluster, rather than computing the similarity between query and each service in the dataset. In our approach, we use the Formal Concept Analysis (FCA) formalism to organize the constructed hierarchical clusters into concept lattices according to their topics. Thus, service discovery may be achieved more easily using the concept lattice. FCA is a mathematical theory that permits the identifications of groups of objects having common attributes [3]. In our work, FCA takes as input a constructed hierarchical tree represented as a formal context and produces the set of all the formal concepts which form a concept lattice. A formal context is denoted by $K = (N, T, I)$ where $N$ is a set of objects (i.e. hierarchical nodes), $T$ is a set of attributes (i.e. Topics), and $I$ is a binary relation between $N$ and $T$ ($I \subseteq N \times T$). $(n, t) \in I$ denotes the fact that object $n \in N$ is in relation through $I$ with attribute $t \in T$ (i.e. Also read as $n$ has $t$). A concept lattice defines a hierarchical representation of objects and attributes, in which a certain concept inherits all the extents (objects) of its descendants and all the intents (attributes) of its ascendants.

We utilized Lattice Miner\(^2\) is a formal concept analysis software tool for the construction, visualization and manipulation of concept lattices via approximation, projection and selection. We generate a concept lattice for each level in the hierarchical tree. The rationale for this is to allow users to customize their queries to find specific or general services. Thus, the service request is compared with each level of clusters to find the best matched service advertisement. Let $Q = \{w_1, w_2, \ldots, w_n\}$ be a user query that contains a set of words $w_i$ produced by a user. At the first time, we represent the user’s query as a distribution over topics. Thus, for each topic $z_f$ we calculate the relatedness between query $Q$ and $z_f$ based on the probability distribution $\phi$ using Equation 3.

$$P(Q|z_f) = \prod_{w_i \in Q} P(w_i|z_f)$$ (3)

From constructed concept lattice, we can retrieve the best cluster by enquiring the lattice using the related topic for user query. Such a query returns a sub-lattice of cluster sharing the specified set of topics, which are highly probable to provide similar functionalities. Then, we extract the related services of these clusters. An abstraction of this mechanism is shown in Figure 2. In our approach, we use the generated probabilities $\theta$ and $\phi$ as the base criteria for computing the similarity between retrieved services and a user’s query. For this, we model information retrieval as a probabilistic query to the topic model. We note this as $P(Q|s_i)$ where $Q$ is the set of words contained in the query. Thus, using the assumptions of the topic model, $P(Q|s_i)$ can be calculated by equation 4.

$$P(Q|s_i) = \prod_{w_k \in Q} P(w_k|s_i) = \prod_{w_k \in Q} \sum_{z_f=1}^{T} P(w_k|z_f)P(z_f|s_i)$$ (4)

The most relevant services are the ones that maximize the conditional probability of the query $P(Q|s_i)$. Consequently, relevant services are ranked in order of their similarity score to the query. Thus, we obtain automatically an efficient ranking of the retrieved services.

**IV. Evaluation**

**A. Web Services Corpus**

Our experiments are performed out based on real-world web services obtained from the WSDL service retrieval test collection called SAWSDL-TC\(^3\). The WSDL corpus consists of 1088 semantically annotated WSDL 1.1-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 different queries (i.e. 42 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. Table II lists the number of services and requests from each domain. Before applying the proposed approach, we deal the WSDL corpus. The objective of this preprocessing is to identify the textual concepts of services, which describe the semantics of their functionalities. WSDL corpus processing consists of several steps: Features extraction, Tokenization, Tag and stop words removal, Word stemming and Service Transaction Matrix construction (See [4] for more details). The observed textual concepts are represented in a Service Transaction Matrix (STM). We use the STM as training data.

---

\(^2\)http://sourceforge.net/projects/lattice-miner/

\(^3\)http://www.semwebcentral.org/projects/sawSDL-tc
for our implementation of the CTM model (based on the Blei’s implementation\(^4\), which is a C implementation of CTM using Variational EM for Parameter Estimation and Inference).

The proposed clustering mechanism based on CTM is evaluated using the FScore measure [18] which treats each cluster as if it was the result of a query for which all the services of the class were the desired set of relevant services (See Section IV-C). We evaluated also our service matchmaking and ranking system by calculating the Precision at n \((P@n)\) and the Normalised Discounted Cumulative Gain \((NDCG_n)\) for the retrieved services obtained for each of the sample queries (See Section IV-D). These are standard evaluation techniques used in IR to measure the accuracy of a search and matchmaking mechanism.

All experiments were performed on a personal computer MacBook Pro with Intel Core 2 Duo processor, 2.4 GHz, and 6 Go 1067 MHz of RAM.

**B. Determining the Number of Topics**

The choice of the number of topics corresponding to the original dataset has an impact on the interpretability of the results. In CTM model, the number of topics must be decided before the training phase. There are several methods to choose the number of topics that lead to best general performance [16]. In computational linguistics, the measure of perplexity has been proposed to assess generalizability of text models. We computed the perplexity of a held-out test set to choose the optimal number of topics. A lower perplexity score indicates better generalization performance. Assume we have \(M \) web services as a held out dataset \(D_{test} \) and each web service \(s\) contains \(N_d\) word tokens. More formally, the perplexity for a dataset \(D_{test}\) is defined by:

\[
Perplexity = \exp \left( - \sum_{d=1}^{M} \sum_{n=1}^{N_d} \log p(w_n|s_d) / \sum_{d=1}^{M} N_d \right) \tag{5}
\]

Where \(p(w_n|s_d)\) is the probability of having word \(w_n\) given the \(d\)-th service.

Figure 3 shows the perplexity of the held-out data for learned model by varying the number of topics (lower numbers are better). As observed from this figure, the better performance is obtained for \(K = 90\) (where \(K\) is the number of topics). Thus, for the evaluation of our service matchmaking and ranking, the CTM model was trained to generate 90 topics.

![Perplexity values obtained for learned CTM model](image)

**Fig. 3.** Perplexity values obtained for learned CTM model

**C. Evaluation of Clustering Quality**

The quality of a hierarchical clustering mechanism based on Correlated Topic Model was measured using a FScore measure that looks at the class labels of the web services assigned to each cluster. We compared the performance of our hierarchical clustering mechanism using different probability metrics (i.e. Section IV-C2).

1) **FScore measure**: FScore measure identifies for each class of services the node in the hierarchical tree that best represents it and then measures the overall quality of the tree by evaluating this subset of clusters [18]. Suppose \(q\) classes represent the partitioned web services (service domains) and \(n\) the total number of services. Given a particular class \(L_i\) of size \(n_i\) and a particular cluster \(C_j\) of size \(n_j\), suppose \(n_{ij}\) services in the cluster \(C_j\) belong to class \(L_i\), then the FScore of this class and cluster is defined to be:

\[
F(L_i, C_j) = \frac{2 \times R(L_i, C_j) \times P(L_i, C_j)}{R(L_i, C_j) + P(L_i, C_j)} \tag{6}
\]

Where \(P(L_i, C_j) = n_{ij} / n_j\) is the precision value and \(R(L_i, C_j) = n_{ij} / n_i\) is the recall value defined for the class \(L_i\) and the cluster \(C_j\). The FScore (Equation 7) of the class \(L_i\), is the maximum FScore reached value at any node in the hierarchical tree \(T\).

\[
FScore(L_i) = \max_{C_j \in T} F(L_i, C_j) \tag{7}
\]

The FScore (Equation 8) of the entire hierarchical tree is defined to be the sum of the individual class specific FScores weighted according to the class size.

\[
FScore = \sum_{i=1}^{q} \frac{n_i}{n} FScore(L_i) \tag{8}
\]

Where \(q\) is the total number of domains in the dataset and \(n\) the total number of services. A perfect clustering solution will

<table>
<thead>
<tr>
<th>#</th>
<th>Domain</th>
<th>Services</th>
<th>Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Communication</td>
<td>58</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Economy</td>
<td>358</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>Education</td>
<td>285</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>Food</td>
<td>34</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>Geography</td>
<td>60</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>Medical</td>
<td>73</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>Military</td>
<td>40</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>Simulation</td>
<td>16</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>Travel</td>
<td>164</td>
<td>6</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1088</strong></td>
<td><strong>42</strong></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE II**

**NUMBER OF SERVICES AND QUERIES FOR EACH DOMAIN**

be the one in which every class has a corresponding cluster containing the same set of services in the resulting hierarchical tree, in which case the FSscore will be one. In general, the higher the FSscore value is, the better the clustering solution is.

2) Probability metrics for hierarchical clustering’s distance measure: In [8], the authors reviewed on ten of the most popular probability metrics/distances used by statisticians and probabilists. We use five metrics and test their influence on our learned hierarchical clustering quality. The selected metrics are: (1) symmetric KL divergence ($D_{skl}$), (2) Hellinger distance ($D_h$), (3) symmetric $\chi^2$ distance ($D_{\chi^2}$), (4) Cosine Distance ($D_{cos}$) and (5) Total Variational Distance ($D_{tvd}$), whose definitions are given below.

$$KL(p, q) = \sum_{i=1}^{V} p_i \log\left(\frac{p_i}{q_i}\right)$$

$$D_{skl}(p, q) = \frac{1}{2}[KL(p, q) + KL(q, p)]$$

$$D_h(p, q) = \left(\sum_{i=1}^{V}\left(\sqrt{p_i} - \sqrt{q_i}\right)^2\right)^{1/2}$$

$$D_{\chi^2}(p, q) = \sum_{i=1}^{V} \frac{(p_i - q_i)^2}{p_i + q_i}$$

$$D_{tvd}(p, q) = \frac{1}{2}\sum_{i=1}^{V} |p_i - q_i|, \quad D_{tvd}(p, q) \in [0, 1]$$

In our experiment, we compared the accuracy of our hierarchical clustering algorithm using different probability metrics, to that of a classical algorithm HAC (Hierarchical Agglomerative Clustering). The 9 service domains described previously (Table II) are used as the base classes to evaluate the FSscore (i.e Equation 8) of the entire hierarchical tree. Figure 4 shows the FSscore values of our constructed hierarchical clustering with different distance measures. As seen in this figure, all distance measures provide the same accuracy and are presented on the same level. The results show that the hierarchical clustering method based on CTM performs significantly than HAC. The Correlated Topic Model allows each service to exhibit multiple topics with different proportions. Thus, it can capture the heterogeneity in grouped data that exhibit multiple latent factors. This makes our hierarchical clustering mechanism based on CTM an ideal solution for web services clustering in large repositories.

D. Evaluation of Service Discovery and Ranking

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$^n$). These are standard evaluation techniques used in IR to measure the accuracy of a search and matchmaking mechanism. For this experiment, we use the services description collected from the test collection. As described previously, the services are divided into nine domains and some service request are provided together with a relevant response set for each query. The relevance sets for each query consists of a set of relevant service 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.

1) Precision@n: 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 14:

$$\text{Precision@n} = \frac{|\text{RelevantServices} \cap \text{RetrievedServices}|}{|\text{RetrievedServices}|}$$

(14)

Where the list of relevant services to a given query is defined in the test collection.

2) Normalised Discounted Cumulative Gain: 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 Information Retrieval for evaluating ranking results. The $NDCG^n$ for $n$ retrieved services is given by Equation 15.

$$NDCG^n = \frac{DCG^n_{\text{ideal}}}{DCG^n_{\text{actual}}}$$

(15)

Where $DCG^n_{\text{actual}}$ is the Discounted Cumulative Gain and $DCG^n_{\text{ideal}}$ is the Ideal Discounted Cumulative Gain. The $DCG^n_{\text{ideal}}$ is found by calculating the $DCG^n$ of the first $n$ returned services. The $DCG^n$ is given by Equation 16.

$$DCG^n = \sum_{i=1}^{n} \frac{\text{relevance}(i)}{\log_2(1 + i)}$$

(16)

Where $n$ is the number of retrieved services and $\text{relevance}(s)$ is the graded relevance of the service in the $i$th 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. \( \text{NDCG}_n \) values vary from 0 to 1.

We evaluated the effectiveness of our Web Service Discovery and Ranking based on CTM and FCA (labelled \( \text{HCTM} \)) by calculating the \( \text{Precision}@n \) and \( \text{NDCG}_n \). The sample queries (42 queries) provided in the dataset are all in the form of SAWSDL documents and contain the semantic requirements together with a text description of the queried functionality. Table III shows an overview of the 9 selected queries (i.e. One query for each domain).

<table>
<thead>
<tr>
<th>ID</th>
<th>Domain</th>
<th>Query Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Communication</td>
<td>Title Comedy Film</td>
</tr>
<tr>
<td>2</td>
<td>Economy</td>
<td>Dvd Player &amp; Mp3 Player Price</td>
</tr>
<tr>
<td>3</td>
<td>Education</td>
<td>Country Skilled Occupation</td>
</tr>
<tr>
<td>4</td>
<td>Food</td>
<td>Grocery Store Food</td>
</tr>
<tr>
<td>5</td>
<td>Geography</td>
<td>Get Altitude Above Sea Level Of Location</td>
</tr>
<tr>
<td>6</td>
<td>Medical</td>
<td>Hospital Investigating</td>
</tr>
<tr>
<td>7</td>
<td>Military</td>
<td>Government Missile Funding</td>
</tr>
<tr>
<td>8</td>
<td>Simulation</td>
<td>Fall Down Pill</td>
</tr>
<tr>
<td>9</td>
<td>Travel</td>
<td>City Country Hotel</td>
</tr>
</tbody>
</table>

**TABLE III**
OVERVIEW OF THE 9 QUERIES USED IN OUR EVALUATION

Our method HCTM described in Section III is compared with a syntax-based approach powered by Apache Lucene\(^5\) (labelled \( \text{ApacheLucene} \)) and also method from the SAWSDL-MX2\(^6\) hybrid semantic matchmaker for SAWSDL services[13]. The number of topics used for CTM was determined by evaluating the perplexity performance of the system for different number of topics (See Section IV-B). For our approach HCTM, these queries are represented in a vector of topics using Equation 3 and then matched to the services in topic space using a mechanism described in Section III. For SAWSDL-MX2, the search queries are submitted using the SAWSDL-MX user interface. For the ApacheLucene approach, the text descriptions taken from the query template are used as the query string.

Generally, the top most relevant retrieved services by a search engine are the main results that will be selected and used by the user. The averaged \( \text{Precision}@n \) and \( \text{NDCG}_n \) were measured for up to the first 30 retrieved services from the complete list of results. In Information retrieval, \( \text{NDCG}_n \) gives higher scores to systems which rank a search result list with higher relevance first and penalizes systems which return services with low relevance.

Our approach HCTM with different concept lattices generated for different levels in hierarchical tree is evaluated by comparing the averaged \( \text{Precision}@n \) and the \( \text{NDCG}_n \) values over all 42 queries. Thus, our method with different concept lattice is labelled \( \text{HCTM Level n} \) (where \( n \) corresponds to the \( n \)-th level in hierarchical tree).

The \( \text{Precision}@20 \) and \( \text{NDCG}@20 \) values are obtained over nine queries for ApacheLucene, SAWSDL-MX2, \( \text{HCTM Level} 1 \). The results are shown in Table IV and V respectively. In both cases, the results show that our method HCTM Level 1 gives a higher \( \text{Precision}@20 \) and \( \text{NDCG}@20 \) for all 9 queries. The ApacheLucene and SAWSDL-MX2 were unable to find the relevant web services for some queries (i.e. Query 2, 5 and 9). The comparison of average \( \text{Precision}@n \) and \( \text{NDCG}_n \) values over all 42 queries are respectively shown in Table 5 and 6. The results show that our methods HCTM with different Levels (i.e. Level 1 and 2) perform better than others. In fact, they give a higher precision values for all requests. 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. We note also that the method HCTM Level 1 performs better than the method HCTM Level 2. This reflects that the retrieved services obtained by the first method (i.e HCTM Level 1) are specific to the user’s query and those obtained by the second method (i.e. HCTM Level 2) are general. ApacheLucene and SAWSDL-MX2 have a low \( \text{NDCG}_n \) because, as shown in the \( \text{Precision}@n \) results, both approaches are unable to find some of the highly relevant services. The HCTM Search methods holds a higher \( \text{Precision}@n \) and \( \text{NDCG}_n \) than ApacheLucene and SAWSDL-MX2 for any number of retrieved services, this reflects the accuracy of the ranking mechanism used by our method.

Table VI shows the average query response times for ApacheLucene, SAWSDL-MX2 and our HCTM method. The HCTM methods give a faster query response time than the other search methods.

\(^5\)http://lucene.apache.org/
\(^6\)http://projects.semwebcentral.org/projects/sawdsl-mx
In our approach, we use the Formal Concept Analysis (FCA) formalism to organize the constructed hierarchical clusters from semantic service descriptions and model the correlation among them. Thus, service discovery may be achieved more easily using the concept lattice. In our experiment, we compared the accuracy of our hierarchical clustering algorithm using different probability metrics, to that of a classical algorithm (HAC (Hierarchical Agglomerative Clustering)). The results show that the hierarchical clustering method based on CTM performs better than HAC. We also evaluated our service discovery approach by calculating the \( \text{Precision}_n \) and \( \text{NDCG}_n \). The comparison of \( \text{Precision}_n \) and \( \text{NDCG}_n \) shows that our HCTM methods with different Levels (i.e. Level 1 and 2) perform better than others (i.e. ApacheLucene and SAWSDL-MX2 Matchmaker). Future work will propose a new interactive discovery method using the sub-lattice that approximate the user’s query, see Fig. 2. The main idea is to use each concept of the resulted sub-lattice to answer the user’s query. Several alternative results became possible and each one may be interpreted as a view-oriented answer.

**V. CONCLUSION**

In this paper, we proposed a non-logic-based matchmaking approach that uses correlated topic model to extract topics from semantic service descriptions and model the correlation between the extracted topics. Based on the topic correlation, service descriptions can be grouped into hierarchical clusters. In our approach, we use the Formal Concept Analysis (FCA) formalism to organize the constructed hierarchical clusters into concept lattices according to their topics. Thus, service discovery may be achieved more easily using the concept lattice. Thus, service discovery may be achieved more easily using the concept lattice. In our experiment, we compared the accuracy of our hierarchical clustering algorithm using different probability metrics, to that of a classical algorithm (HAC (Hierarchical Agglomerative Clustering)). The results show that the hierarchical clustering method based on CTM performs better than HAC. We also evaluated our service discovery approach by calculating the Precision at \( n \) and NDCG at \( n \). The comparison of Precision at \( n \) and NDCG at \( n \) shows that our HCTM methods with different Levels (i.e. Level 1 and 2) perform better than others (i.e. ApacheLucene and SAWSDL-MX2 Matchmaker). Future work will propose a new interactive discovery method using the sub-lattice that approximate the user’s query, see Fig. 2. The main idea is to use each concept of the resulted sub-lattice to answer the user’s query. Several alternative results became possible and each one may be interpreted as a view-oriented answer.

**REFERENCES**


<table>
<thead>
<tr>
<th>Method</th>
<th>Times (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HCTM Level 1</td>
<td>297</td>
</tr>
<tr>
<td>HCTM Level 2</td>
<td>518</td>
</tr>
<tr>
<td>ApacheLucene</td>
<td>1163</td>
</tr>
<tr>
<td>SAWSDL-MX2</td>
<td>3045</td>
</tr>
</tbody>
</table>

**TABLE VI**

AVERAGED QUERY RESPONSE TIMES