Probabilistic Matchmaking Methods for Automated Service Discovery

Gilbert Cassar, Payam Barnaghi, *Senior Member, IEEE,* and Klaus Moessner, *Senior Member, IEEE*

Abstract—Automated service discovery enables human users or software agents to form queries and to search and discover the services based on different requirements. This enables implementation of high-level functionalities such as service recommendation, composition, and provisioning. The current service search and discovery on the Web is mainly supported by text and keyword based solutions which offer very limited semantic expressiveness to service developers and consumers. This paper presents a method using probabilistic machine-learning techniques to extract latent factors from semantically enriched service descriptions. The latent factors are used to construct a model to represent different types of service descriptions in a vector form. With this transformation, heterogeneous service descriptions can be represented, discovered, and compared on the same homogeneous plane. The proposed solution is scalable to large service datasets and provides an efficient mechanism that enables publishing and adding new services to the registry and representing them using latent factors after deployment of the system. We have evaluated our solution against logic-based and keyword-based service search and discovery solutions. The results show that the proposed method performs better than other solutions in terms of precision and normalised discounted cumulative gain values.

Index Terms—Computer Web Services, discovery, machine-learning, probabilistic methods, LDA.

1 Introduction

The ubiquitous access and significant advantages of *Service-Oriented Computing* (SOC) is driving businesses to implement or transform their online applications into *Web Services* (WS) [1]. Consequently, the Internet is shifting from data and web applications to a framework of data and service platforms [2]–[4]. However the common technologies used to describe, publish, and discover these services offer very little expressiveness to service developers. The *World Wide Web* lacks a homogeneous structure (beyond that of the service interface) for describing and discovering functional and non-functional parameters in service descriptions. This results in different ways to name and define parameters and describe internal processes which inhibits straightforward integration of WS [5].In this paper we focus on search and matchmaking functionalities to support automated service discovery for software agents and human users.

1.1 Motivation

Automated service discovery is an important aspect in service oriented computing as many high-level service oriented concepts such as service composition, service provisioning, and service recommendation highly rely on the precision of an underlying automated service discovery engine. To achieve such high-level goals, service discovery needs first to extend from syntax-based matchmaking and take an automated approach where machines interpret the meaning behind the service description data and matchmaking is performed based on both functional and non-functional attributes of a service. The services used in service oriented computing technologies are software artifacts that are autonomous, self-described, reusable, and highly portable [6]. Such artifacts are searched for based on their functionality (*e.g.* inputs, outputs, processes, and operations) rather than based on their text descriptions. Semantic service discovery has an edge over syntax-based approaches and most state-of-the-art service discovery approaches are now based on semantic service description models [7]–[21]. Work on semantic service discovery is split in three main categories: logic-based approaches, non-logic-based approaches, and hybrid approaches.

Logic-based semantic service discovery approaches [10]–[13] use a reasoner to infer new knowledge from the concepts and relationships defined in semantic service descriptions. Logical reasoning tends to be very accurate given its solid mathematical basis [8]. These kind of approaches provide an improvement on the short-comings of syntax-based methods but come at the expense of increased complexity. A known limitation of logic-based approaches is that when two concepts are semantically synonymous but defined differently in their terminological definitions, the similarity

• *G. Cassar, P. Barnaghi, and K. Moessner are with the Centre for Communication Systems Research, University of Surrey, Guildford, GU2 7XH, UK.*
E-mail: {g.cassar, p.barnaghi, k.moessner}@surrey.ac.uk

This paper describes work undertaken in the context of the EU IoT-A project, IoT-A: Internet of Things - Architecture (http://www.iot-a.eu/public) contract number: 257521. The second and third authors are also funded by the EU ICT Iot.est project (www.Ict-Iot.est.eu) contract number: 288385.

between the two is not captured by the subsumption hierarchy and a reasoner would fail to find the match between the two [18]. Another limitation is the complexity of logic-based discovery solutions which makes logical reasoning and the discovery process over large service repositories an intractable process.

Non-logic-based semantic service discovery approaches [5], [14]–[16] aim to reduce the complexity of semantic matchmaking by analysing the frequency of occurrence of certain terms within service descriptions and determine semantics which are implicit in service descriptions. These approaches generally use techniques such as graph matching, linguistic analysis, data mining, and information retrieval (IR) [8]. Approaches based on information retrieval techniques are very popular as these techniques require the service descriptions to be expressed in terms of vectors which make it easier to use mathematical tools to process the data. However, this transformation results in the loss of the machine-interpretable semantics found in some service descriptions and thus non-logic-based approaches cannot perform the fine grained matchmaking that is possible with logic-based approaches.

Hybrid Matchmakers [17]–[23] combine the advantages of Non-Logic-based techniques with the fine grained reasoning capabilities of Logic-based techniques. Klusch *et al.* [18] state that the objective of this hybrid semantic matchmaking is to appropriately exploit both crisp logic-based and non-logic-based semantic retrieval where using each of the solutions alone could fail.

Although literature suggests that hybrid matchmakers always outperform syntax-based and logic-based matchmakers in terms of precision and ranking, hybrid matchmakers suffer from interoperability problems similar to logic-based methods. This happens because the algorithm for logic-based matchmaking differs for various service description models and thus to make a logic-based or hybrid matchmaker compatible with different service descriptions models, a separate implementation of the logic matchmaking component is required for each service description model.

In this paper, we propose a non-logic-based matchmaking method that uses machine learning techniques and in particular *Probabilistic Latent Semantic Analysis* (PLSA) [24], [25] and *Latent Dirichlet Allocation* (LDA) [26] to extract latent factors from semantic service descriptions and search for services in latent factor space where heterogeneous service descriptions are all represented as a probability distribution over latent factors. The latent factors are discussed in more detail in Section 5. This approach reduces the complexity of semantic service matchmaking while still preserving the knowledge in the machine-interpretable semantics.

PLSA and LDA are unsupervised machine-learning techniques that use a generative probabilistic model to map high-dimensional count vectors (such as the distribution of semantic concepts describing the services in a repository) to a lower dimensional representation in latent factor space. Representing the information contained in service descriptions in terms of latent factors rather than semantic concepts reduces the dimensionality of service descriptions as each latent factor represents a group of concepts. Latent factors can also resolve problems in service discovery related to synonymy (different concepts referring to the same meaning). Dimensionality reduction also decreases the number of computation steps needed to compare services thus simplifying the matchmaking process. Heterogeneous service descriptions (*i.e.;* service descriptions represented using different service description models such as OWL-S[1] and WSMO[2]) are all converted to the same latent factor space thus creating a homogeneous plane on which these different service descriptions can be compared with each other directly in terms of latent factors. Queries are also expressed in terms of latent factors (using a technique known as query folding-in [27] described in Section 5.3). Automated service discovery is then obtained by matching the queries and services in latent factor space using vector proximity measures. The contributions of our work include:

1) The service descriptions are transformed from a number of semantic annotations to a vector of smaller dimensions (latent factors). Services are classified and matched based on these vectors, thus the dimensions of the required computations are reduced.
2) An efficient mechanism for publishing new service descriptions using a technique called *Folding-In* [27]. Any new service description, as long as the service description technology or defined parameters are used in the initial training set, can be transformed into latent factor space and included in the index repository.
3) Interoperability between different service description technologies and different methods for defining parameters is achieved by mapping all the service descriptions to a latent factor space (using folding-In).
4) The comparisons of Precision @ n and Normalised Discounted Cumulative Gain ($NDCGn$) values for LDA with full registry search, LDA with nearest three clusters search, PLSA with full registry search, the Hybrid Matchmaker OWLS-M4, the logic-based matchmaker OWLS-M0, and a syntax-based search indicate that the methods based on LDA presented in this paper outperform all the other matchmakers in terms of ranking of the most relevant services.

1. http://www.w3.org/Submission/OWL-S/
2. http://www.wsmo.org/

2 Proposed System

The work in this paper builds upon our previous work on representing service descriptions in terms of latent factors [28], [29]. Latent factors are a concept introduced by Probabilistic Topic models [30] (sometimes also called Latent Factor Models). These are a family of generative probabilistic graphical models based on the assumption that documents are generated by a mixture of topics (latent factors) where topics are probability distribuitions on terms. Based on this assumption, Bayesian inference can be used to invert this process and infer the unobserved hidden topics (latent factors) that generated a collection of documents. Different probabilistic Topic models have been used to analyse the meaning of words in documents [24]–[26], [31]–[33]. In our work, a latent factor is associated with a group of semantic concepts that can appear in service descriptions and can be expressed as a probability distribution over semantic concepts $P(c|z)$.

A service description is generated from a topic model by following a number of probabilistic sampling rules that describe how the concepts in the service description are generated by sampling from a set of latent factors. First a distribution of latent factors for the service description is chosen. Then a latent factor from the chosen distribution is picked randomly and a concept is drawn from that latent factor. The chosen concept is put in the service description and the process is repeated for every concept in the service description. This kind of generative process is based on the *bag-of-words assumption* [30] and assumes the concepts in a service description appear in a random manner (*i.e.* no assumption is made about the order of the concepts as they appear). The only matter of relevance to this model is the number of times a concept is produced in a service description.

As a result, service descriptions can be described in terms of a vector $s = \{z_1, z_2, ..., z_K\}$, where every dimension z_k is an integer between 0 and 1 describing the probability of the service being related to latent factor k. This vector is essentially the probability distribution over latent factors for the service. This reduces the dimensionality of service descriptions and allows service descriptions written in different heterogeneous service description models to be represented on the same homogeneous plane (*i.e.* Latent Factors; see Section 5 for details).

We assume there exists a registry containing a number of semantic service descriptions. The concepts in the service descriptions are the observed data and the challenge is to find the best set of latent factors that hypothetically generated the observed dataset. This statistical inference problem is to find the probability distribution over concepts associated with each latent factor and the latent factor distribution for each service description.

In this context, we provide our descriptions and main evaluations based on the *Onthology Web Language for Services* (OWL-S) model; a Service Description Model that provides both rich expressive descriptions and well-defined semantics. OWL-S has been used as the base service model for evaluating various works on semantic service matchmaking [18]–[21]. However, as mentioned earlier, the proposed solution is not limited to only one technology. In the following sections we elaborate this concept in more detail.

Semantic concepts are extracted from the OWL-S service descriptions using a reasoner and the occurrence of each semantic concept in each service description is tallied in a *Service Transaction Matrix* (discussed further in Section 4). The Service Transaction Matrix provides the training set (observed data) that is used as input for the machine-learning techniques that we use to infer the latent factors.

The learned probability distribution over concepts for each latent factor can be used to compute the probability distribution of latent factors for any new service description using a technique called *Folding-in* [30] (discussed further in Section 5.3). Folding-in can be used on service descriptions written in any semantic service description model as long as concepts used to describe the service have already been observed in the initial training dataset. Folding-in can also be used to calculate the probability distribution of latent factors for a service request, provided that the service request contains machine-interpretable semantic concepts.

3 The Aspect Model for Semantic Service Descriptions

The Aspect Model is a generative topic model developed by Hoffman *et. al.* [24], [25] to model the probabilistic sampling rules that dictate how terms in a text document are generated by sampling terms from hidden variables (topics). This model belongs to a family of topic models [24]–[26], [31]–[33] which assume that text documents are generated from a mixture of topics, where a topic consists of a probability distribution over words.

Semantic service descriptions are different from text documents because they usually contain very little textual descriptions, include concepts (described by URIs) instead of words, and are full of property assertions in the form of subject-predicate-object expressions. We adapt the Aspect Model to define how semantic documents such as OWL-S service descriptions are generated by sampling from a set of latent factors $\{z_1, z_2, ..., z_K\}$. The model assumes that semantic concepts observed in a service description are independent of each other given the latent factors they were generated from. No assumptions are made about the order in which the concepts appear in

a service description. However, the property assertions in semantic service descriptions cannot be ignored. The components making a property assertion; subject-predicate-object, are clearly not independent from one another and cannot be assumed to have been generated from different latent factors. If we assume that the components of a property assertion (subject-predicate-object) are independent from each other, all the logic relationships expressed in machine-interpretable semantics would be lost when the latent factors are inferred from the dataset of service descriptions. A different approach is required in order to preserve the semantic data when projecting semantic service descriptions to a different mathematical plane. In [20] the concepts from OWL-S service descriptions are extracted and represented in a collection of Fuzzy Multisets. The advantage of this approach is that the role of a concept in a service description as an input, output, precondition, or effect is also captured in the conversion.

In order to preserve semantic relationships, in our approach we assume that the *subject*, *predicate*, and *object* in a property expression were generated by the same latent factor and are not independent of each other. For example; from the service description *EBookOrder1* shown in Figure 1, the concept *UserAccount* which is described by the property *hasInput* is assumed to be a single concept *hasInput_UserAccount* that was generated by sampling from a single latent factor. Using this approach, a service in which *UserAccount* is defined as an input will be distinguished from a service in which *UserAccount* is defined as an output.

The latent factors are initially unknown and statistical inference is used to find the latent factors that best describe the observed semantic concepts. To fit the model, we use two machine-learning algorithms: *Expectation Maximization* [24] for PLSA and *Collapsed Gibbs Sampling* [34] for LDA. A training set of service descriptions is required for these algorithms to compute the best fit of latent factors that explain the observed semantic concepts in the service descriptions. Both algorithms are unsupervised learning algorithms and do not require any labelled data as input or any intervention from the user.

The key probability distribution that we are interested in learning from the Aspect Model is the probability distribution of concepts for each latent factor $P(c|z_k)$. This probability distribution can be used to estimate the distribution of latent factors for any new service description using a technique called *Folding-in* [30] (discussed further in Section 5.3).

4 The Service Transaction Matrix

A *Service Transaction Matrix* (STM) is an adaptation of the method used in Information Retrieval for representing the information contained in a distributed database. This conversion facilitates statistical analysis of the data. The STM has as many rows as the number of services in the repository and as many columns as the number of concepts in the dataset. Each row represents a service s as a vector of N dimensions where each dimension represents the occurrence of a concept in the description of that service. A vector $\bar{s_i}$ describing a service i is denoted as:

$$\bar{s_i} = \{c_1, c_2, ..., c_N\}, \qquad where \; \forall \; c \; \epsilon \; \mathbb{Z}^+ \qquad (1)$$

Different strategies exist for extracting such concepts from service descriptions and converting them to vector form. The *Text Frequency and Inverse Text Frequency* (TF/IDF) algorithm is used in [35] to represent a dataset of the WSDL service descriptions in an STM. A variation of TF/IDF is proposed in [5] where a higher weight is given to the IDF value. This approach aims to normalise the bias of the TF measure as the frequency of concepts in very short documents such as service descriptions tends to be incidental. One limitation of these methods is that they treat service descriptions like text documents and do not take advantage of the semantic annotations used in the service descriptions.

In our approach concepts and their roles are extracted from the OWL-S service descriptions using OWL API[3] and Pellet[4]. Stop words in the text descriptions of the concepts are removed using the Stanford Log-Linear POS-tagger[5]. The array of concepts extracted for sample service *EBookOrder1* is shown below:

EBookOrder1 = {e-book, order, web, service,

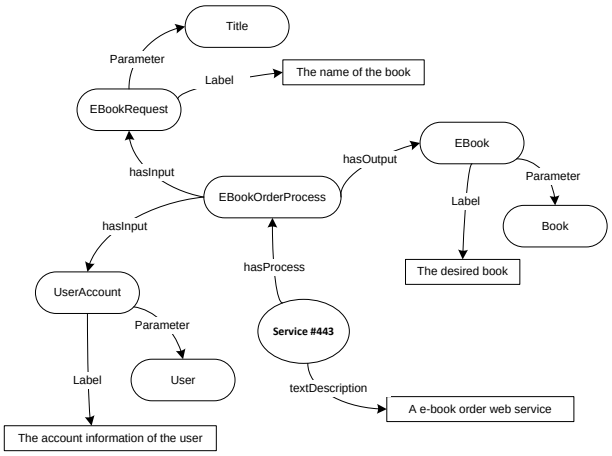

Fig. 1. A sample OWL-S service description (EBookOrder1).

3. http://owlapi.sourceforge.net/
4. http://clarkparsia.com/pellet/
5. http://nlp.stanford.edu/software/tagger.shtml

desired, book, account, information, user, hasProcess_EBookOrderProcess, hasInput_UserAccount, hasInput_User, hasInput_EBookRequest, hasInput_Title, hasOutput_EBook, hasOutput_Book}

The STM is created after parsing the whole repository using our concept extraction mechanism. Each entry c_{ij} of the matrix represents the number of times concept j occurs in service i where $\forall\ c_{ij} \epsilon\ \mathbb{Z}^+$.

5 Extracting Latent Factors

In this section we discuss two unsupervised machine-learning techniques that we use to extract latent factors from the service descriptions. The first method we discuss is Probabilistic Latent Semantic Analysis (PLSA); a generative statistical model used for analysing co-occurrence of data. The second method is Latent Dirichlet Allocation (LDA) which improves the performance of PLSA by introducing a Dirichlet prior on the distribution of latent factors over service descriptions [26].

5.1 Probabilistic Latent Semantic Analysis

PLSA is an unsupervised machine-learning technique that maps high-dimensional count vectors (such as the ones expressed in the STM) to a lower dimensional representation in *Latent Factor Space* [24]. PLSA is based on the Aspect Model; a latent variable model that associates an unobserved class variable $z_k \epsilon \{z_1, z_2, ..., z_K\}$ with each observation [36].

PLSA discovers a hidden dimension behind the vector of concepts describing a service, *i.e.* topics that include concepts in the service descriptions. The concepts are observable variables and their occurrence in a service description can be described in a vector as defined by Equation 1. Topics on the other hand are latent factors which are not directly observable through examining a service description. These latent factors are learned through statistical inference. A set of services can then be described as a multinomial probability distribution $P(z|s)$.

The distribution $P(z|s)$ is a matrix with K rows and M columns. Where K is the number of generated latent factors and M is the number of service descriptions. Each entry z_{ki} represents the probability of service i belonging to topic k. Each service description s_i can be described as an array of latent factors denoted as:

$$\bar{s_i} = \{z_{1i}, z_{2i}, ..., z_{Ki}\}, \qquad \forall\ z\ \epsilon\ \mathbb{R}^+ \leq 1 \tag{2}$$

Representing a service in terms of these latent variables reflects the likelihood of a service belonging to certain concept groups [35].

The joint probability of the observed concept c_j in service description s_i is denoted as:

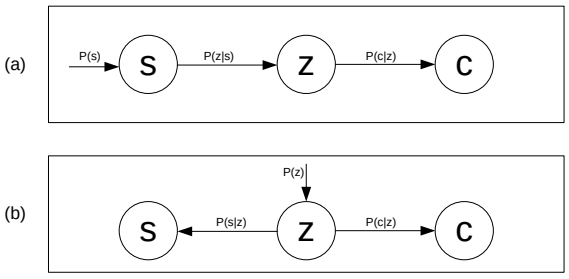

Fig. 2. Graphical model representation of the PLSA model in asymmetric (a) and symmetric (b) parametrization.

$$P(s, c) = P(s)\, P(c|s) \tag{3}$$

and by assuming that a service and a concept are conditionally independent given a set of K latent factors, we can express $P(c|s)$ in terms of latent factors:

$$P(c|s) = \sum_{k=1}^{K} P(z_k|s)\, P(c|z_k) \tag{4}$$

The graphical model representation of this conditional independence is shown in Figure 2(a). This model indirectly associates the concepts to their corresponding service descriptions by introducing an intermediate layer of latent factors. The model achieves dimensionality reduction by mapping a high-dimensional $P(s, c)$ space (describing services in terms of concepts) into a lower K-dimensional latent factor space (describing services in terms of latent factors) [35]. By substituting equation 4 in equation 3, we obtain an equivalent representation of the model given by Equation 5 (illustrated in Figure 2(b)).

$$P(s, c) = \sum_{k=1}^{K} P(z_k)\, P(s|z_k)\, P(c|z_k) \tag{5}$$

The parameters $P(z)$, $P(s|z)$, and $P(c|z)$ can be found using a model fitting technique such as the *Expectation Maximization* (EM) algorithm as described in [24]. The PLSA model for our approach is implemented using the PennAspect[6] model which uses maximum likelihood to compute the three parameters: $P(c|z)$, $P(d|z)$, and $P(z)$. In our work, half of the dataset is used to train the algorithm and the other half is used for validation in order to prevent overfitting [27].

5.2 Latent Drichlet Allocation

Latent Dirichlet Allocation (LDA) is another machine-learning technique which uses a generative probabilistic model for collections of discrete data. LDA

6. http://www.cis.upenn.edu/ũngar/Datamining/software_dist/PennAspect/index.html

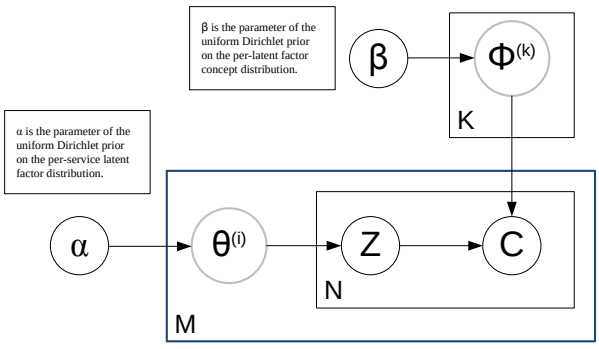

Fig. 3. Plate Representation of LDA Model.

introduces a Dirichlet prior on the $P(z|s)$ distribution in order to simplify the problem of statistical inference [26]. The principle of LDA is similar to PLSA: mapping high-dimensional count vectors to a lower dimensional representation in latent factor space.

The generative process of LDA is shown in Figure 3 where the plates represent replicates. The outer plate represents service descriptions (repeated M times) and the inner plate represents the repeated choice of latent factors and concepts within a service description (repeated N times) where M is the number of service descriptions and N is the number of concepts. The concepts c represent the observed data and z represents the latent variables which need to be estimated. ϕ_k is a multinomial distribution of concepts for latent factor k. θ_i is a multinomial distribution of latent factors for service i. α and β are constant hyperparameters of the Dirichlet priors on ϕ_k and θ_i respectively.

Using the same notation described in PLSA, the generative model of LDA can be represented as:

$$P(c_j) = \sum_{k=1}^{K} P(c_j|z_k)\, P(z_k) \tag{6}$$

where $P(z_k)$ is the probability that latent factor k is sampled for concept j and $P(c_j|z_k)$ is the probability of sampling concept j given latent factor k.

The multinomial distributions with dirichlet priors are defined as:

$$\Phi^{(k)} = P(c|z) \tag{7}$$

and

$$\Theta^{(i)} = P(z) \tag{8}$$

Instead of estimating $P(s|z)$ and $P(c|z)$ as in PLSA, the LDA generative model estimates Φ, Θ, and z. Different methods can be used to train the algorithm and estimate these parameters. Blei *et al.* [26] use variational inference with the Expecatation Maximization algorithm. Wang *et al.* [27] estimate the parameters using a method based on Gibbs Sampling which was proposed in [33] and [30]. In general these solutions provide methods using Expectation Maximisation to estimate Φ, Θ, and z and do not effect the function of the LDA model.

In our work, the LDA model is implemented using LingPipe[7] toolkit. This toolkit uses Gibbs sampling to train the algorithm and estimate the parameters Φ, Θ, and z.

5.3 Folding-In

Folding-in is a technique used for publishing (*i.e.* fitting) new service descriptions into the latent factor model after the model has been trained. New services and also service requests can be expressed in terms of latent factors by computing the distribution of latent factors for the concepts that describe a new service or a service request.

For PLSA, once the algorithm is trained and the parameters are found, any new service description or request can be folded into the model using [35]:

$$P(z_k|s_{new}) = \frac{P(s_{new}|z_k) \cdot P(z_k)}{\sum_{j=1}^{K} P(s_{new}|z_j)} \tag{9}$$

For LDA, after training the algorithm, new service descriptions or queries can be folded in using Gibbs sampling by assuming a fixed service description to concept probabilities $P(c|s)$ and sampling the assignment of concepts to latent-factors in the new service description or request [27].

5.4 Determining the Number of Latent Factors

In PLSA and LDA the number of latent factors must be decided before training. The choice of the number of latent factors with respect to the original dataset has an impact on the interpretability of the results. A solution with too few latent factors will result in distributions over concepts for each latent factor that are too broad [30]. A solution with too many latent factors will result in uninterpretable latent factors. Griffiths and Styvers [37] discuss a method for estimating the posterior probability of the model while integrating over all possible parameter settings. The number of latent factors is chosen based on the model that leads to the highest posterior probability.

Topic models based on non-parametric bayesian statistics have also been used to automatically determine the number of latent factors that best explains the observed data [38], [39].

Another method is to empirically determine the number of latent factors that leads to the best general

7. http://alias-i.com/lingpipe/

performance [30]. Griffiths and Styvers [37] discuss that as the number of latent factors is increased the model can more accurately fit the data until an optimal point is reached. Increasing the number of latent factors beyond this point makes the model more complex than necessary and results in fitting noise (*i.e.* overfitting) which degrades the performance of the model. We evaluated the performance of our system for increasing numbers of latent factors and the results peak at $K = 90$ for 1000 services (where K is the number of latent factors) before the performance starts to decrease. These evaluation results are shown in Figures 7 and 8. In Section 9 we discuss this in more detail and describe the evaluation settings.

6 Probabilistic Service Matchmaking and Ranking

To achieve more automated service discovery, the structure of a service request also needs to move from syntax-based to machine-interpretable semantics. Traditional service request methods require the user to specify the request in the form of a string. These methods rely on syntax and are harder for a machine to interpret the meaning behind a service request. Moreover, it is difficult for the user to specify functional requirements using a string. Machine-interpretable semantics provide an alternative to service request methods. Rather than specifying the requirements in a string, the user can fill-in a service request template rich with machine-interpretable semantics.

A semantic service request template follows the same structure as a service description model. It can contain the definitions of the required IO interface and also other semantic concepts that describe the internal processes that are required. The use of machine interpretable semantics enables the user to specify functionalities (using existing semantic technologies) and also the roles of the concepts involved.

Filling-in a service request template can be a manual, semi-automated, or fully-automated process. A semi-automated process uses machine-interpretable semantics to recommend concepts to the user as the request template is being filled-in. A fully automated process occurs in machine-to-machine interactions where the request template is filled automatically by the client machine. The template does not need to be filled-in completely. The user is free to input only as much information as desired. However, a more detailed service request template yields more accurate and relevant search results.

Automated service discovery engines need to provide a search interface that enables both human and machine clients to submit a request in the form of a template which defines what type of functionality is needed. The service discovery engine compares the service descriptions to the request and returns services which match the functionalities described in the request. This ensures that services returned by the discovery engine have inputs, outputs, and processes that can fulfil the requirements of the client.

Our service search and matchmaking mechanism works by computing the degree of match between a query and a service description in latent factor space. We map queries into latent factor space using the folding-in techniques described in Section 5.3. The degree of match between the probability distribution of latent factors for the request and a service description can be calculated using a vector similarity measure. This is possible because for any service description/request the probability distribution over K latent factors are expressed as a vector $\bar{p} = \{z_1, z_2, ..., z_K\}$ where each dimension z_k represents the probability of that service description/request being generated by sampling from latent factor k. We use a vector similarity measure called *Multidimensional Angle* (or *Cosine Similarity*). It uses the cosine of the angle between two vectors. Multidimensional angle is used in various approaches dealing with vector-space analysis of service data such as [15], [16], and [35]. This proximity measure is computationally efficient because if a dimension is not present in both vectors that are being compared, it will be automatically dropped from the calculation.

The multidimensional angle between a vector containing the distribution of latent factors p of a service and a vector containing the distribution of latent factors q of a query can be calculated using Equation 10.

$$cos(p, q) = \frac{p \cdot q}{\|p\| \cdot \|q\|} = \frac{\sum_{i=1}^{f} p_i q_i}{\sqrt{\sum_{i=1}^{f} p_i^2 \sum_{i=1}^{f} q_i^2}} \quad (10)$$

where f is the number of latent-factors.

The multidimensional angle takes values in the interval $[0, 1]$ where 0 indicates no similarity and 1 indicates identical vectors. Using this degree of match, a query can be compared to all services in the registry. This also allows ranking the results based on the similarity score in descending order.

7 Probabilistic Clustering

Comparing a service request to all service descriptions stored in a registry can be computationally expensive in large service repositories. In such cases, it is desirable to have a scheme that helps us reduce the scope of our search and thus reduce the amount of comparisons required. By organising the service registry into clusters, we can determine which cluster contains information that is most similar to a service request and restrict the scope of our search to that cluster. This reduces the number of comparisons required to answer a request but it also means that any relevant services that weren't assigned to the clusters we look into will be missed.

The work described in this section extends our previous work on probabilistic service clustering [28]. We use the latent factors learned from the probabilistic models (PLSA and LDA) to group the services into clusters. Each latent factor generated by the model represents a particular subset of concepts. After the model is trained, the distribution of concepts for each latent factor is known and all the services in the dataset can be described as a distribution of latent factors (*i.e.* a vector $\bar{p} = \{z_1, z_2, ..., z_K\}$ where each dimension z_k reflects the probability of that service description being generated by sampling from latent factor k). We create K clusters; where K is the number of generated latent factors (*i.e.* a cluster for each latent factor). The vector of latent factors describing each service is used to determine which latent factor best describes the service. The service is then assigned to the cluster corresponding to that latent factor. An abstraction of this mechanism is shown in Figure 4. If a service has more than one latent factor that is related to it, the service will be assigned to each of the clusters that correspond to these latent factors.

This approach gives us a number of advantages over classical clustering algorithms [28]. The dimensionality of the model is reduced as all services can be described in terms of a small number of latent factors rather than a large number of concepts. The algorithm is also more scalable and can be applied to large datasets because only a small portion of the data set is required to train the algorithm. The rest of the service descriptions and any other new service published to the repository can be folded-in and assigned to clusters easily without high computational requirements. Consequently, searching for a service inside a cluster can be performed by searching for matching latent factors rather than matching the text describing the service to a set of key words extracted from the service request.

Additionally, a service could be assigned to multiple clusters (for example the three best fitting clusters). This will increase the scope of each search. Multiple cluster assignments achieve higher search accuracy. However, it comes at the cost of increased number of comparisons and computations (See evaluation results in Section 9.6.

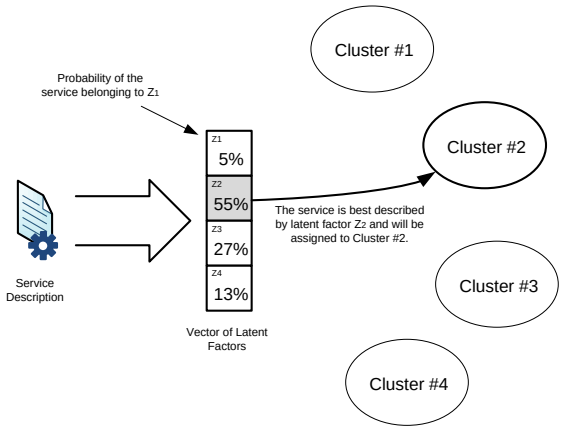

Fig. 4. Abstraction of the clustering mechanism for four generated Latent Factors.

8 Extending to Bigger Service Environments

In order to extend LDA and PLSA to massive service environments, the probability distribution of latent factors over concepts needs to be re-learned when the number of new concepts that are introduced for describing the services significantly increases. In a situation where many previously unseen semantic concepts start appearing in new service descriptions, the model would have to be re-trained in order to be able to recognise the new concepts. The method can scale up by re-training the machine-learning model whenever the new concepts exceed a proportional segment of existing concepts. This can be defined by heuristics and depends on preferred trade-off between freshness and computational efficiency. The learning process is unsupervised and can be set to be done as an off-line task when the number of new concepts and services are significantly high. The model can be re-trained offline and once the new LDA or PLSA model is constructed it can replace the existing model. This means that even if the training process starts taking longer as the size of the repository increases, the performance of the system would still not be hindered by the training phase. When the training phase is completed and the new latent factors are known, the service indexes can be updated and service descriptions containing the previously unseen concepts can now be recognised by the indexing and retrieval processes. By repeating this process, the system can efficiently adapt to new concepts as the repository grows bigger over time.

Other possible solutions including non-parametric Bayesian models such as the Chinese Restaurant Process [40] could be also used to learn new latent factors as new semantic concepts are observed. These models allow the number of latent factors to increase gradually as new concepts are observed without the need of re-learning the latent factors [38], [39]. This would allow latent factor models to be extended more easily to bigger service environments. However, exploring the usage of these models in automated service discovery is beyond the scope of the current paper and will be conducted in our future work.

9 Evaluation

The dataset of service descriptions used in this experiment is obtained from the OWL-S service retrieval test

collection called OWLS-TC v3.0[8]. The dataset consists of 1007 service descriptions defined in OWL-S form. The services are divided into seven categories and a total of 29 sample service requests are provided together with a relevant answer set for each request. Table 1 shows the number of services and requests belonging to each of the seven categories in the dataset.

TABLE 1
Number of Services and Requests for each domain.

Domain	Services	Requests
Education	284	6
Food	34	1
Medical	73	1
Travel	165	6
Communication	58	2
Economy	359	12
Weapon	40	1

Concepts and their roles are extracted from these OWL-S service descriptions using the parser described in Section 4. The observed concepts are represented in a Service Transaction Matrix. The service transaction matrix is used as training data for our implementation of the PLSA model (based on the PennAspect[9] model that uses maximum likelihood to fit the model to the observed data) and our implementation of the LDA model (based on the LingPipe[10] toolkit that uses Gibbs sampling to fit the model to the observed data). In order to prevent overfitting half of the service transaction matrix is used to train the algorithm and the other half is used for validation [27].

The clustering mechanisms based on PLSA and LDA are evaluated using the *Normalised Mutual Information* (NMI) [41] which reflects the accuracy of the clustering scheme against the number of clusters generated. If the number of clusters is increased more than necessary, the NMI value will stop increasing, thus reflecting the fact that no further accuracy is achieved by increasing the number of clusters. The NMI gives an approximation of the required number of latent factors needed to efficiently represent all the data in the repository. There are seven service categories defined in OWLS-TC: communication, economy, education, food, medical, travel, and military. The categories are used as the base classes to evaluate *Purity*, *Entropy*, and *Mutual Information* [41] of the clustering schemes which are in turn used to calculate the NMI.

We evaluated our matchmaking and ranking approach by calculating the *Precision at n* ($P@n$) [42] and the *Normalised Discounted Cumulative Gain* ($NDCG_n$) [43] for the results obtained for each of the sample service requests. These are standard evaluation techniques used in *Information Retrieval* to measure the accuracy of a search mechanism with respect to completeness of the returned results [27]. The averaged Precision at n and Normalised Discounted Cumulative Gain were measured for up to the first 40 services retrieved from the complete list of results.

The methods (based on PLSA and LDA) described in Section 6 are compared with a syntax-based approach powered by Apache Lucene[11] and also methods from the OLWS-MX 2.0[12] semantic Web service matchmaker (M0 and M4) [18]. M0 is a logic-based approach and M4 is a hybrid approach that uses both logic and non-logic-based methods. The number of latent factors used for LDA and PLSA was determined by evaluating the Precision at n and Normalised Discounted Cumulative Gain performance of the system for different number of latent factors. In the next section, the method based on PLSA is labelled *PLSA*, the method based on LDA with search narrowed to the nearest cluster is labelled *LDA with n Cluster Assignments* (where n corresponds to the maximum number of clusters a service can be assigned to), and the method based on LDA with a search which spans the whole dataset is labelled *LDA Full Registry Search*.

The sample service requests provided in the dataset are all in the form of OWL-S templates and contain the semantic requirements together with a text description of the queried functionality. For PLSA and LDA, these request templates are converted to latent factor space using folding in and then matched to the services in latent factor space. For the two OWLS-MX variants the search templates are submitted using the OWLS-MX user interface. For the syntax-based approach, the text descriptions taken from the request templates are used as the query string.

We also investigated how assigning the services to different number of clusters at the same time effects the performance of the search and ranking mechanism. By assigning all service descriptions to more than one cluster, purity becomes a confusing measure because each cluster will now contain service descriptions from a wider variety of categories rather than a very specialised set. LDA with different number of cluster assignments is evaluated by comparing the averaged *Precision at n* ($P@n$) and the *Normalised Discounted Cumulative Gain* ($NDCG_n$) values over all 29 service requests for different numbers of cluster assignments.

All experiments were carried out on a computer with Intel(R) Core(TM)2 Duo T7500 2.2GHz CPU, 2GB RAM, and running Microsoft Windows 7 x86.

9.1 Purity

The Purity of clusters is used as a measure to evaluate the accuracy of a clustering technique [44], [35]. If the pool of services used to evaluate the algorithm

8. http://www.semwebcentral.org/projects/owls-tc/
9. http://www.cis.upenn.edu/ũngar/Datamining/software_dist/PennAspect/index.html
10. http://alias-i.com/lingpipe/
11. http://lucene.apache.org/
12. http://semwebcentral.org/projects/owls-mx/

were originally organised in an ideal set of classes $c = \{c_1, c_2, ..., c_m\}$, then for clusters generated by the algorithm $Z = \{z_1, z_2, ..., z_K\}$ the purity of a clustering algorithm can be computed as:

$$Purity = \frac{1}{M} \sum_{f=1}^{k} max_c \left\{ n_f^c \right\} \tag{11}$$

where M is the total number of services and n_k^c is the number of services in cluster z_k belonging to class c while c varies from 1 to m.

It is easy to obtain a high value of cluster purity if the data set is clustered into a large number of clusters (in relation to the number of available services in the dataset). With a large number of clusters, small clusters will be formed. If each cluster is very small, the likelihood of having a high percentage of the cluster belonging to one known class could be very high.

9.2 Entropy

Entropy is a measure of the consistency for clustering [44]. The entropy for a cluster set $Z = \{z_1, z_2, ..., z_K\}$ is defined as:

$$H(Z) = -\sum_{k=1}^{K} \frac{|z_k|}{M} log \frac{|z_k|}{M} \tag{12}$$

where M is the total number of services and $|z_k|$ is the number of services in cluster z_k.

9.3 Normalised Mutual Information

Mutual Information measures the mutual dependence between two variables. In this context, mutual information for clustering set $Z = \{z_1, z_2, ..., z_K\}$ and an ideal set of classes $c = \{c_1, c_2, ..., c_m\}$ is defined as:

$$I\left(Z, C\right) = \sum_{k=1}^{K} \sum_{i=1}^{m} \frac{|z_k \cap c_i|}{M} log \frac{M\,|z_k \cap c_i|}{|z_k|\,|c_i|} \tag{13}$$

where M is the total number of services.

We use Normalised Mutual Information (NMI) as another measure to evaluate the clusters. Normalised Mutual Information [41] is designed to penalize clustering methods which use a large number of clusters. NMI can show clearly that once a certain number of clusters is reached, no further advantage is gained by increasing the number of clusters. The NMI of a clustering technique can be computed as:

$$NMI\left(Z\right) = \frac{I\left(Z, C\right)}{\left[H\left(Z\right) + H\left(C\right)\right]/2} \tag{14}$$

NMI normalises the Mutual Information with the denominator $\left[H\left(Z\right) + H\left(C\right)\right]/2$ which increases as the number of clusters increases and each cluster becomes smaller. This gives us a mean to compare the quality of a clustering technique for different values of generated clusters. The value of NMI is always between 0 and 1 [41].

9.4 Precision @ n

Precision measure is used to evaluate the results of the search and matchmaking process. Precision @ n [42] is a measure of the precision of the system taking into account the first n retrieved services. Precision reflects the number of retrieved services which are relevant to the search. The precision for a set of retrieved services is given by:

$$precision = \frac{|\{RelevantServices\} \cap \{RetrievedServices\}|}{|\{RetrievedServices\}|} \tag{15}$$

where the set of relevant services to a given request is defined in the high relevance OWLS-TC v3.0 test collection. Only services in the dataset with a graded relevance of 3 are considered for this evaluation.

9.5 Normalised Discounted Cumulative Gain

The $NDCG_n$ [43] is a measure that takes into account the graded relevance of each service retrieved. This measure is particularly useful for evaluating ranking strategies since not all services in a relevance set are of the same relevance to the request. The $NDCG_n$ for n retrieved services is given by Equation 16.

$$NDCG_n = \frac{DCG_n}{IDCG_n} \tag{16}$$

where DCG_n is the Discounted Cumulative Gain and $IDCG_n$ is the Ideal Discounted Cumulative Gain.

The $IDCG_n$ is found by calculating Discounted Cumulative Gain of the ideal first n returned services for a given request. The DCG_n is calculated by Equation 17.

$$DCG_n = \sum_{i=1}^{n} \frac{2^{label(i)} - 1}{log_b(1+i)} \tag{17}$$

where n is the number of services retrieved, $label(i)$ is the graded relevance of the service in the ith position in the ranked list, b is the *Discounting Factor* which models the user's persistence (*e.g.* impatient: $b = 2$; persistent: $b = 14$).

$NDCG_n$ gives higher scores to solutions which rank services with higher relevance first and penalizes solutions which return services with low relevance. In our experiments we set $b = 2$ and used graded relevance scheme with values from 3 (high relevance) to 1 (low relevance). We have used both Precision @ n and $NDCG_n$ for evaluating the matchmaking and ranking results as described in the following section.

9.6 Results

The evaluation results for Entropy, Mutual Information, and NMI of PLSA based clustering and LDA based clustering are shown in Figure 5 and Figure 6. As the numbers of generated latent factors (clusters) is increased, the purity of both clustering mechanisms keeps increasing (as explained in Section 9.1). On the other hand, NMI penalizes the score if the number of clusters is increased beyond a point where generating more clusters does not increase the accuracy of the clustering mechanism [41]. NMI provides a guideline for the number of latent factors that we need to generate in order to obtain an accurate representation of the original dataset. However, for the service search and matchmaking process, services are not just assigned to a cluster but they are also assigned to a vector which describes the distribution of latent factors for each service. Therefore the ideal number of latent factors needed for the whole search and matchmaking process differs from the ideal number required for just clustering the services. Empirical results (Figures 7 and 8) suggest that 90 latent factors lead to the best general performance of the ranking and matchmaking process on the dataset used for evaluation. Thus for the evaluation of the service matchmaking and ranking, the LDA and PLSA models were trained to generate 90 latent factors.

Figure 9 and Figure 10 show the comparison of $P@n$ and $NDCG_n$ scores for LDA with different number of cluster assignments. In both cases, LDA with full registry search represents the best case scenario where the process checks every service in the registry. The LDA methods with different cluster assignments show the effect of restricting the scope of search on the average $P@n$ and $NDCG_n$ scores of the method. LDA with one cluster assignment exhibits the least $P@n$ and $NDCG_n$ performance while as we allow services to be assigned to more clusters (thus increasing the scope of the search), the $P@n$ and $NDCG_n$ performance start approaching that of LDA with full registry search.

The average $P@n$ and $NDCG_n$ are obtained over all 29 service requests for LDA, LDA with 3 Cluster Assignments, PLSA, Text-Matching, OWLS-M0, and OWLS-M4. The results are shown in Figures 11 and 12 respectively. The $P@n$ results show that Text-Matching and the logic-based OWLS-M0 were unable to find some of the relevant services that were not directly related to the queries through logic descriptions or keywords. The PLSA model does not capture the information in the latent factors as efficiently as LDA and thus the search and matchmaking mechanism based on PLSA exhibits poor precision. LDA captured more information in the latent factors than PLSA and the LDA based mechanisms exhibited better precision. LDA with 3 Cluster Assignments performed better than Text-Matching and the logic-based OWLS-M0

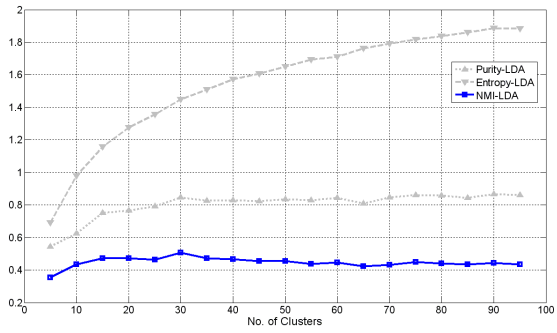

Fig. 5. Purity, Entropy, and NMI for LDA

but exhibited less precision than the LDA Full Registry Search. The LDA Full Registry Search managed to find some of the relevant services that LDA with 3 Cluster Assignments missed out due to the limited scope of the latter mechanism. OWLS-M4 also found more relevant services than the Text-Matching approach and the logic-based OWLS-M0. The LDA Full Registry Search performed better than OWLS-M4 and the other matchmaking mechanisms.

$NDCG_n$ evaluates the ranking mechanism and it is the most important measure to evaluate the automated search and matchmaking process. The top most relevant (*e.g.* the first five or ten) results retrieved by a search and matchmaking process are the main results that will be used by the client user. Both LDA based mechanisms perform better than the other search and matchmaking mechanisms in this experiment. The LDA Full Registry Search holds a higher $NDCG_n$ than all other methods for any number of services retrieved, this reflects the accuracy of the probability based ranking mechanism used by our method. Text-Matching and OWLS-M0 have a low $NDCG_n$ because, as shown in the $P@n$ results, both mechanisms are unable to find some of the highly relevant services. PLSA exhibited poor $NDCG_n$ results as expected due to its inaccuracy in extracting latent factors. OWLS-M4 exhibited a high $NDCG_n$ but was outperformed by both LDA methods for the first five services retrieved and outperformed throughout by the LDA Full Registry Search.

Table 2 shows the average query response times for LDA, LDA with different number of cluster assignments, PLSA, and Text-Matching. The average query response time of the OWLS-MX variants could not be obtained due to an unsolved software issue on the available open source software[13]. Text-Matching is faster than the other methods as it is powered by the optimised text search engine Apache Lucene. PLSA is faster than the LDA methods because the folding-

13. By accessing the OWLS-MX software, some queries take extremely long time to respond which made them inapplicable for comparison. This was also acknowledged by the developers of the software

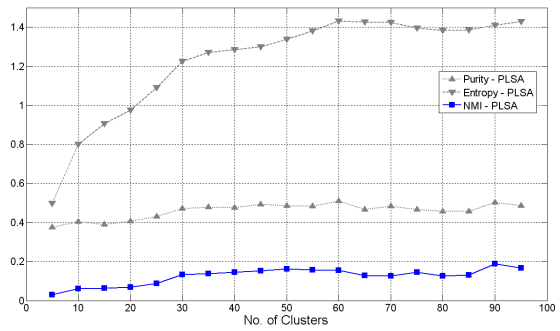

Fig. 6. Purity, Entropy, and NMI for PLSA

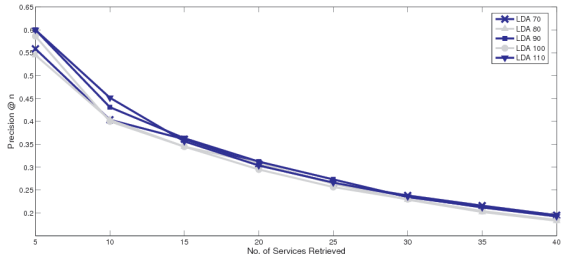

Fig. 7. $P@n$ peformance of LDA for different numbers of Latent Factors.

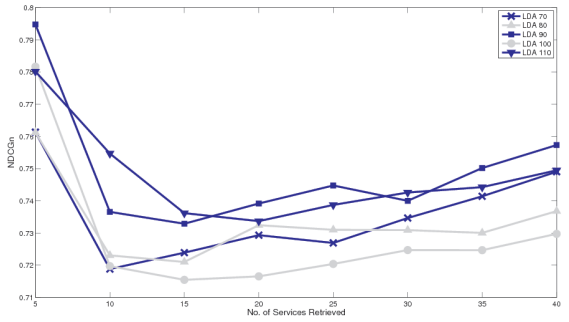

Fig. 8. $NDCG_n$ peformance of LDA for different numbers of Latent Factors.

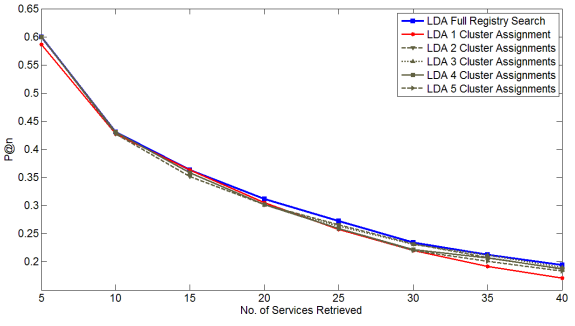

Fig. 9. Comparison of P@n scores for LDA with different numbers of Cluster Assignments for each service.

in process requires less computations. The LDA with

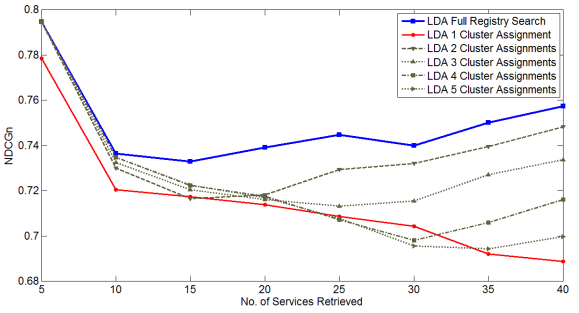

Fig. 10. Comparison of NDCGn scores for LDA with different numbers of Cluster Assignments for each service.

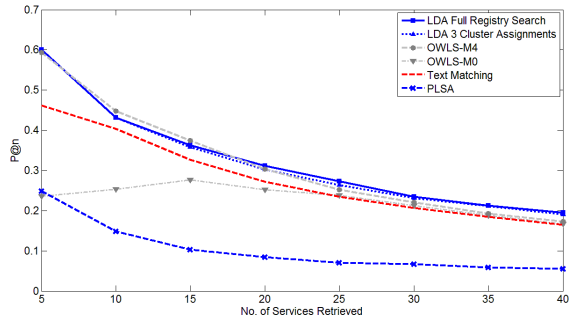

Fig. 11. Averaged $P@n$ values

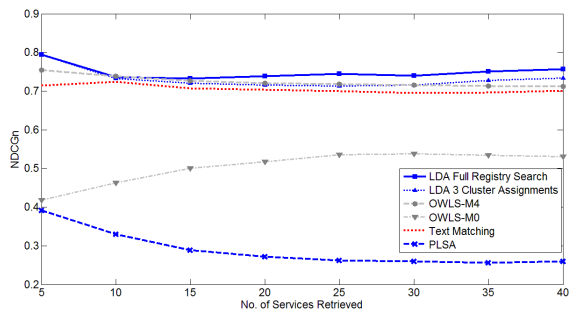

Fig. 12. Averaged $NDCG_n$ values

clustering methods gives a faster query response time than the LDA Full Registry Search. However, this comes at the cost of reduced accuracy in matchmaking and ranking. Increasing the number of cluster assignments increases the accuracy of the matchmaking and ranking but also increases the query response time as more services need to be checked before the results are found. Considering the fact that query folding-in will happen independent from underlying dataset, the response time for LDA is still highly scalable. The process takes longer than PLSA or Text-Matching due to initial query folding in. However, for similarity measures the process relies on a vector similarity function (which in this work is a Cosine Similarity measure) to match every service in the repository with

the request. Therefore if n is the number of services in the repository, this makes the complexity of the main process as O(n) plus an initial query fold in process time. For the LDA with Clustering search, the time scale will remain lower as only a limited number of clusters will be searched for each query and it is mainly dependent on how many services will be included in each cluster.

TABLE 2
Averaged Query Response Times.

Method	Time (ms)
LDA Full Registry Search	15.0743
LDA 1 Cluster Assignment	11.7316
LDA 3 Cluster Assignments	12.3868
LDA 5 Cluster Assignments	13.4581
Text Matching	10.6350
PLSA	10.7271

10 Conclusions and Future Work

This paper describes using probabilistic machine learning for service matchmaking and ranking. We discuss using LDA and PLSA methods to transform the service descriptions to latent factor space. The proposed solution applies unsupervised probabilistic machine learning methods (i.e. LDA and PLSA) to the service description data and creates a lower dimensional vector model to represent the services. A fold-in approach is used to process service search queries and to add new services to the model. A vector distance model is used to calculate the similarity of vector representations in the latent factor space. We have evaluated our results against a syntax-based search (by employing Apache Lucene) and also existing logic-based and hybrid methods using OWLS-MX software. The results show that our LDA-based approach performs better than other solutions in terms of Precision@n and Normalised Discounted Cumulative Gain ($NDCG_n$) values. The proposed solution is also scalable to large service repositories as it does not require re-training of the model when new services are added. The new services can be folded into the model by using Gibbs Sampling or any other similar methods. The similarity value between services is obtained by measuring the distance between service vectors or service and request vectors in the latent factor space. The similarity value is also used as a notation for similarity ranking. The proposed methods enable automated service discovery by processing service request templates or keyword/attributes-based queries and retrieving the most relevant services to the submitted request. The latent factor model can also be used for clustering services according to their similarity to a set of specific clusters in the latent factor space. The clusters can be used for distribution of service descriptions in the latent factor space among different registries in a distributed environment for large-scale service platforms. By applying this method similarity of a service request to different clusters can be used as criteria to reduce the scope of the search to a limited number of clusters.

The future work will focus on including a semantic analysis and logic based search in combination with our current solution to further enhance the search results. This will use the current model to find the service similarities based on the clusters and then the detailed search will be performed based on a semantic analysis method. We will also investigate using the service discovery and matchmaking solution to support an automated service composition process. Enhancements of setting the number of latent factors and number of clusters using different methods will also be considered in the future work to optimize the result and processes in LDA and PLSA based solutions.

References

[1] A. R. Hevner, D. Zhang, and J. L. Zhao, "Guest editorial: Introduction to the special issue on modeling and implementation of service enterprise systems," *Services Computing, IEEE Transactions on*, vol. 3, no. 2, pp. 86 –88, 2010.

[2] Q. Yu, X. Liu, A. Bouguettaya, and B. Medjahed, "Deploying and managing web services: issues, solutions, and directions," *The VLDB Journal*, vol. 17, pp. 537–572, May 2008.

[3] M. Huhns and M. Singh, "Service-oriented computing: key concepts and principles," *Internet Computing, IEEE*, vol. 9, no. 1, pp. 75 – 81, 2005.

[4] G. Zheng and A. Bouguettaya, "Service mining on the web," *IEEE Trans. Serv. Comput.*, vol. 2, pp. 65–78, January 2009.

[5] A. Segev and E. Toch, "Context-based matching and ranking of web services for composition," *IEEE Transactions on Services Computing*, vol. 99, no. PrePrints, pp. 210–222, 2009.

[6] U. Sehgal, K. Kaur, and P. Kumar, "The anatomy of a large-scale hyper textual web search engine," in *Proceedings of the 2009 Second International Conference on Computer and Electrical Engineering - Volume 02*, ser. ICCEE '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 491–495.

[7] R. Nayak and B. Lee, "Web service discovery with additional semantics and clustering," in *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 555–558.

[8] K. Mohebbi, S. Ibrahim, M. Khezrian, K. Munusamy, and S. G. H. Tabatabaei, "A comparative evaluation of semantic web service discovery approaches," in *Proceedings of the 12th International Conference on Information Integration and Web-based Applications; Services*, ser. iiWAS '10. New York, NY, USA: ACM, 2010, pp. 33–39.

[9] M. Klusch, "Chapter 4: Semantic web service coordination," in *CASCOM: Intelligent Service Coordination in the Semantic Web*, 2008.

[10] W. Fang, L. Moreau, R. Ananthakrishnan, M. Wilde, and I. Foster, "Exposing uddi service descriptions and their metadata annotations as ws-resources," in *Proceedings of the 7th IEEE/ACM International Conference on Grid Computing*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 128–135.

[11] K. Iqbal, M. L. Sbodio, V. Peristeras, and G. Giuliani, *Semantic Service Discovery using SAWSDL and SPARQL*. Washington, DC, USA: IEEE Computer Society, 2008.

[12] T. Pilioura and A. Tsalgatidou, "Unified publication and discovery of semantic web services," *ACM Trans. Web*, vol. 3, pp. 11:1–11:44, July 2009.

[13] S. Parsa and K. Fakhr, "Using agent-oriented reasoning engine and sdc graph for optimizing semantic web services discovery," in *Computer Conference, 2009. CSICC 2009. 14th International CSI*, 2009, pp. 423 –430.

[14] H. Fethallah, C. Amine, and B. Amine, "Automated discovery of web services: an interface matching approach based on similarity measure," in *Proceedings of the 1st International Conference on Intelligent Semantic Web-Services and Applications*, ser. ISWSA '10. New York, NY, USA: ACM, 2010, pp. 13:1–13:4.
[15] O. Mola, P. Emamian, and M. Razzazi, "A vector based algorithm for semantic web services ranking," *Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on*, pp. 1 –5, apr. 2008.
[16] C. Platzer, F. Rosenberg, and S. Dustdar, "Web service clustering using multidimensional angles as proximity measures," *ACM Trans. Internet Technol.*, vol. 9, no. 3, pp. 1–26, 2009.
[17] M. Klusch and F. Kaufer, "Wsmo-mx: A hybrid semantic web service matchmaker," *Web Intelli. and Agent Sys.*, vol. 7, no. 1, pp. 23–42, 2009.
[18] M. Klusch, B. Fries, and K. Sycara, "Automated semantic web service discovery with owls-mx," in *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, ser. AAMAS '06. New York, NY, USA: ACM, 2006, pp. 915–922.
[19] S.-L. Pan and Y.-X. Zhang, "Ranked web service matching for service description using owl-s," *Web Information Systems and Mining, 2009. WISM 2009. International Conference on*, pp. 427 –431, nov. 2009.
[20] G. Fenza, V. Loia, and S. Senatore, "A hybrid approach to semantic web services matchmaking," *Int. J. Approx. Reasoning*, vol. 48, pp. 808–828, August 2008.
[21] M. Klusch and P. Kapahnke, "isem: Approximated reasoning for adaptive hybrid selection of semantic services," in *Semantic Computing (ICSC), 2010 IEEE Fourth International Conference on*, sept. 2010, pp. 184 –191.
[22] M. Klein and B. Knig-ries, "Coupled signature and specification matching for automatic service binding," in *In Proc. of the European Conference on Web Services (ECOWS 2004*. Springer, 2004, pp. 183–197.
[23] U. Keller, R. Lara, H. Lausen, A. Polleres, and D. Fensel, "Automatic location of services," ser. Lecture Notes in Computer Science, vol. 3532. Springer Berlin / Heidelberg, 2005, pp. 1–16.
[24] T. Hofmann, "Probabilistic latent semantic analysis," in *Proc. of Uncertainty in Artificial Intelligence, UAI99*, 1999, pp. 289–296.
[25] ——, "Unsupervised learning by probabilistic latent semantic analysis," *Machine Learning*, vol. 42, pp. 177–196, 2001, 10.1023/A:1007617005950.
[26] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003.
[27] W. Wei, P. Barnaghi, and A. Bargiela, "Probabilistic topic models for learning terminological ontologies," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, pp. 1028–1040, 2010.
[28] G. Cassar, P. Barnaghi, and K. Moessner, "Probabilistic methods for service clustering," in *International Semantic Web Conference*. Springerlink, 2010.
[29] ——, "A probabilistic latent factor approach to service ranking," in *Intelligent Computer Communication and Processing (ICCP), 2011 IEEE International Conference on*, aug. 2011, pp. 103 –109.
[30] M. Steyvers and T. Griffiths, *Latent Semantic Analysis: A Road to Meaning*. Laurence Erlbaum, 2007, ch. Probabilistic topic models.
[31] T. L. Griffiths and M. Steyvers, "A probabilistic approach to semantic representation," in *In Proceedings of the 24th annual conference of the*, 2002.
[32] ——, "Prediction and semantic association," in *Advances in Neural Information Processing Systems*. MIT Press, 2003, p. 15.
[33] ——, "Finding scientific topics," *Proceedings of the National Academy of Sciences*, vol. 101, no. Suppl. 1, pp. 5228–5235, April 2004.
[34] J. Liu, "The collapsed gibbs sampler in bayesian computations with application to a gene regulation problem." *Journal of the American Statistical Association*, vol. 89, pp. 958–956, 1994.
[35] J. Ma, Y. Zhang, and J. He, "Efficiently finding web services using a clustering semantic approach," in *CSSSIA '08: Proceedings of the 2008 international workshop on Context enabled source and service selection, integration and adaptation*. New York, NY, USA: ACM, 2008, pp. 1–8.
[36] T. Hofmann, J. Puzicha, and M. I. Jordan, "Learning from dyadic data," in *Proceedings of the 1998 conference on Advances in neural information processing systems II*. Cambridge, MA, USA: MIT Press, 1999, pp. 466–472.
[37] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *PNAS*, vol. 101, no. suppl. 1, pp. 5228–5235, 2004.
[38] D. M. Blei, T. L. Griffiths, M. I. Jordan, and J. B. Tenenbaum, "Hierarchical topic models and the nested chinese restaurant process," in *Advances in Neural Information Processing Systems*. MIT Press, 2004, p. 2003.
[39] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, "Hierarchical dirichlet processes," *JASA*, vol. 101, 2006.
[40] D. M. Blei, T. L. Griffiths, and M. I. Jordan, "The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies," *J. ACM*, vol. 57, pp. 7:1–7:30, February 2010.
[41] C. D. Manning, P. Raghavan, and H. Schtze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.
[42] C. Buckley and E. M. Voorhees, "Evaluating evaluation measure stability," in *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '00. New York, NY, USA: ACM, 2000, pp. 33–40.
[43] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, Oct. 2002.
[44] B. Mandhani, S. Joshi, and K. Kummamuru, "A matrix density based algorithm to hierarchically co-cluster documents and words," in *WWW '03: Proceedings of the 12th international conference on World Wide Web*. New York, NY, USA: ACM, 2003, pp. 511–518.

Gilber Cassar is a Research Fellow in the Centre for Communication Systems Research (CCSR) at the University of Surrey. His research interests include Internet of Things, Web services, service composition, machine learning, and probabilistic graphical models,

Payam Barnaghi is a Lecturer in the Centre for Communication Systems Research (CCSR) at the University of Surrey. His research interests include machine learning, Internet of Things, semantic Web, Web services, information centric networks, and information search and retrieval. He is a senior member of IEEE.

Klaus Moessner is a Professor for Cognitive Networks and Services, in the Centre for Communication Systems Research at the University of Surrey, UK. Klaus earned his Dipl-Ing (FH) at the University of Applied Sciences in Offenburg, Germany, an MSc from Brunel University and PhD from the University of Surrey (UK). His research interests include dynamic spectrum allocation, cognitive networks, reconfiguration management, service platforms and adaptability of multimodal user interfaces.