A Hybrid Semantic Matchmaker for IoT Services

Gilbert Cassar, Payam Barnaghi, Wei Wang, Klaus Moessner
Centre for Communication Systems Research
University of Surrey,
Guildford, UK, GU2 7XH
{g.cassar, p.barnaghi, wei.wang, k.moessner} @surrey.ac.uk

***Abstract*—The use of semantic Web technologies and service oriented computing paradigm in Internet of Things research has recently received significant attention to create a semantic service layer that supports virtualisation of and interaction among "Things". Using service-based solutions will produce a deluge of services that provide access to different data and capabilities exposed by different resources. The heterogeneity of the resources and their service attributes, and dynamicity of mobile environments require efficient solutions that can discover services and match them to the data and capability requirements of different users. Semantic service matchmaking process is the fundamental construct for providing higher level service-oriented functionalities such as service recommendation, composition, and provisioning in Internet of Things. However, scalability of the current approaches in dealing with large number of services and efficiency of logical inference mechanisms in processing huge number of heterogeneous service attributes and metadata are limited. We propose a hybrid semantic service matchmaking method that combines our previous work on probabilistic service matchmaking using latent semantic analysis with a weighted-link analysis based on logical signature matching. The hybrid method can overcome most cases of semantic synonymy in semantic service description which usually presents the biggest challenge for semantic service matchmakers. The results show that the proposed method performs better than existing solutions in terms of precision ($P@n$) and normalised discounted cumulative gain ($NDCG_n$) measurement values.**

I. Introduction

Transparent and seamless access to millions of smart devices and resources is one of the main promises of the *Internet of Things* (IoT) [1]. *Service Oriented Architecture* (SOA) is a promising solution for enabling access to smart devices through loosely coupled services [2]. Web Services provide interfaces for emerging technologies in pervasive systems and make the capabilities and/or data of physical or software entities available to other entities as services on the Web. A service-oriented approach allows resources such as sensors, actuators, and other mobile devices to be represented as a Web service, providing common interfaces that allow users or machines to access their functionality or data through the Internet. We term the services exposed by the connected Things in the physical world as *IoT Services*. Adding semantics to IoT services makes this information machine-interpretable and enables more autonomous IoT service interactions.

Semantic service matchmaking is an important and challenging task in distributed service-oriented environments such as the IoT because it enables human users or software agents to form queries and to search and discover IoT services based on different requirements. This enables implementation of high-level functionalities such as IoT service recommendation, composition, and provisioning. A common practice in semantic service matchmaking is to take advantage of machine-interpretable annotations in the service descriptions to match the semantic *input/output* (IO) signature of a service to a service request [3]. Methods based on logical reasoning tend to be very accurate given its solid mathematical basis. However, strictly matching the semantic signature of an IoT service to the request alone may lead to false negatives when services are relevant to the request but do not have an IO signature that directly matches that of the request [4]. Another known limitation of logic-based approaches is that when two concepts are semantically synonymous but defined differently in their terminological definitions, the similarity between the two is not captured by the subsumption hierarchy and a reasoner would fail to find the match between the two [5].

The limitations of logic-based semantic service matchmakers gave rise to a separate category of non-logic-based semantic matchmakers. Non-logic-based semantic service discovery approaches [6], [7], [8], [9] 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) [10] to process the meta-data provided in service descriptions in terms of vectors. 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 as logic-based approaches. Furthermore non-logic-based semantic matchmakers do not possess the logic-based functions to determine whether the IO parameters of a service are compatible with the requirements of the user.

Hybrid semantic matchmakers [4], [5], [11], [12], [13], [14], [15] combine the advantages of Non-Logic-based techniques with the fine grained reasoning capabilities of Logic-based techniques. Klusch *et al.* [5] state that the objective of this hybrid semantic matchmaking is to appropriately exploit both crisp logic-based and non-logic-based semantic matchmaking where using each of the solutions alone could fail.

We propose a hybrid semantic service matchmaking method for IoT services that combines our previous work on probabilistic service matchmaking using latent semantic analy-

sis [16] with a logical signature matchmaking method based on the concept of individual *Links* between a source parameter and a destination parameter (defined in Section IV-B). This method provides an added flexibility needed when searching for candidate IoT services to be used in complex mechanisms such as IoT service composition or IoT service provisioning. The hybrid method can overcome most cases of semantic synonomy in IoT service description which usually present the biggest challenge for semantic service matchmakers. The evaluation results show that the proposed method performs better than existing solutions in terms of precision ($P@n$) and normalised discounted cumulative gain ($NDCG_n$) measurement values.

The rest of this paper is structured as follows. Section II describes the IoT service modelling framework. In Section III, we briefly explain our previous work on probabilistic service matchmaking. Section IV-B presents the use of *links* and a weighted-link measure for matching the IO signature of an IoT service to a request. Section V discusses how the probabilistic service matchmaking and the weighted-link measure can be combined to create a hybrid semantic matchmaker. Sections VI and VII perform a comparative study between our method and existing semantic service matchmakers and describe the evaluation results respectively. We discuss the merits and limitations of our method and describe the future work in Section VIII.

II. The Service Description Model

Semantic service modelling provides a machine interpretable framework for representing many aspects (*e.g.*, functional, non-functional and transactional attributes) of services. The semantic Web service community has developed several models for semantically describing general Web services such as the Ontology Web Language for Services (OWL-S)[1] and Web Service Modelling Ontology (WSMO)[2]. The work in [17] proposed the 'Entity-Device-Resource' model for representing IoT resources and services based on the Semantic Sensor Network ontology [18]. However, these heavyweight and complex models are not suitable for describing IoT services. IoT services exposed by IoT resources mostly have limited computation capabilities and often operate in dynamic and constrained physical environments; therefore, they are far less reliable and stable compared to the carefully designed and maintained Web services. Their logic is much simpler and their output usually represents observation and measurement of features of interest of physical entities (therefore, service models have to be associated with IoT resources). Despite these characteristics, in a service oriented IoT, they also need to participate in service composition and the issues on effective service adaptation and compensation mechanisms become prominent.

For these reasons, a semantic IoT service representation model preferably needs to be lightweight to facilitate computation (experiences in ontology design shows that well-designed lightweight ontologies have the potential to be widely adopted), in particular efficient service discovery, composition and adaptation given the stunning number of IoT resources and services. The service model should be associated with the model of its exposing resource and provide constructs for linking to concepts in domain knowledge base (*e.g.*, Geonames ontology[3]) or the linked data[4]. We have developed a lightweight description ontology for IoT service based on the existing research and the aforementioned requirements (See Figure 1).

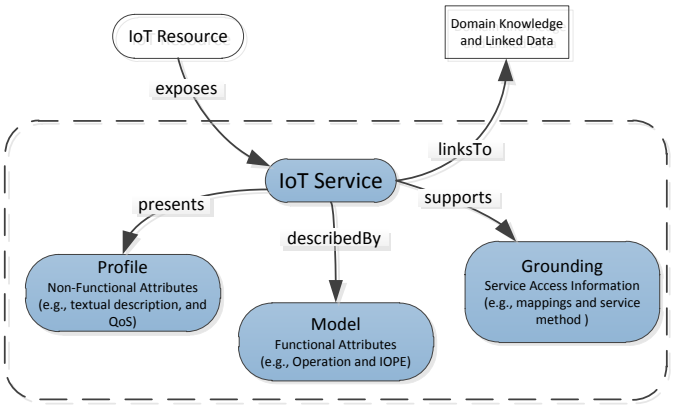

Fig. 1. Overview of the lightweight IoT service description model.

The service model is also designed to be independent of any particular service technologies (*i.e.*, SOAP/WSDL and RESTful services) based on the analysis of their commonalities and distinctiveness. The OWL-S model for SOAP/WSDL services is designed using the 'Profile-Process-Grounding' pattern and much of the complexity stems from the process modelling. On the contrary, the hREST model [19] for RESTful service is too simple: it does not include a profile and grounding which are important for service discovery and access. Our service description model represents a trade-off between these two: being lightweight and service technology independent while at the same time providing sufficient modelling constructs for represent service on the IoT (Details of the ontology can be found at: http://purl.oclc.org/net/unis/IoT.est-Service.owl). We refer to the design pattern as 'Profile-Model-Grounding': Profile and Grounding are adapted from the OWL-S and refined (so it can also be used for RESTful services); the Model excludes the process modelling and is based on the atomic service modelling in OWL-S and RESTful service modelling in hREST. Another advantage of our service model is that although it is not fully compatible with existing modelling methods, it can be easily transformed to each other using a simple program. It should be noted that many of the existing works on semantic service matchmaking [4], [13] and [16] are based on the OWL-S model. For evaluation and comparison purposes, we use a dataset of OWL-S service descriptions

[1]http://www.w3.org/Submission/OWL-S/
[2]http://www.wsmo.org/
[3]http://www.geonames.org/ontology/
[4]http:// linkeddata.org/

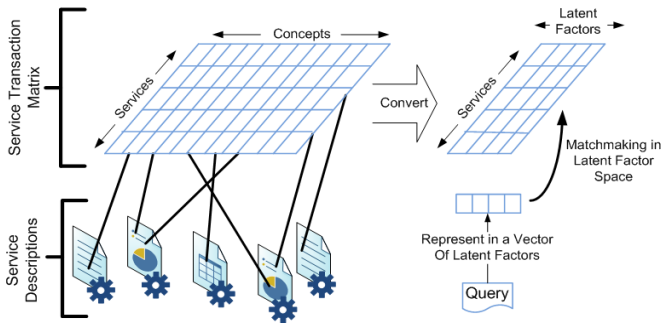

Fig. 2. An abstract representation of the probabilistic service matchmaking.

in this work. The main focus of our work is to show how service descriptions can be used to efficiently discover IoT services in a distributed environment (irrespective of what service description model is used).

III. Probabilistic Service Matchmaking

The work in this paper builds upon our previous work on service search and matchmaking and the ranking of search results [16]. In [16] we showed how semantic concepts can be extracted from OWL-S service descriptions and mapped into a latent factor space using a technique called *Latent Dirichlet Allocation* (LDA) [20]. LDA is an unsupervised machine-learning technique which uses 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 variable space. An abstract overview of our approach is shown in Figure 2.

For every service description s_i the model associates unobserved latent factors $z_1, z_2, ..., z_k$ with the probability of concept c_j appearing in s_i. The generative model of LDA can be represented as:

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

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 LDA model assumes that the probability distributions $P(c|z)$ and $P(z)$ follow a Dirichlet distribution: $\Phi^{(k)} = P(c|z)$ and $\Theta^{(i)} = P(z)$ respectively.

Concepts describing functional parameters and profile data are extracted from OWL-S service descriptions using a reasoner and listed in a *Service Transaction Matrix* which represents the probability distribution P(s,c) of concepts c over service descriptions s. Using the observed data from the service transaction matrix, the parameters Φ and Θ can be estimated using a method based on Gibbs Sampling described in [21]. This algorithm was implemented using the LingPipe[5] toolkit.

The learned model describes each service as a vector of latent factors (as shown in Figure 2). Using this method, a request R (also following the OWL-S model) containing semantic definitions can be converted into latent factor space and matched accurately to the services in a service registry by computing the vector similarity of each service vector to the request vector. We compute this similarity using the proximity measure called *Multidimensional Angle* (also known as *Cosine Similarity*); a measure which uses the cosine of the angle between two vectors [9]. 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 2.

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

where f is the number of latent-factors.

In this work, we use the probabilistic matchmaking method to search for the list of services that most accurately match to a request. The list of search results returned by the probabilistic matchmaking is then passed on to the logical signature matchmaking method (described in the next section) which ranks the services more accurately by checking the compliance of their IO signature with the IO signature of the request.

IV. Logical Signature Matchmaking

Logical signature matching has been used in different works to verify whether the IO parameters of a service are compatible with the IO parameters of a request [3]. A common approach to logical signature matchmaking is to define a set of rules (filters) which dictate what kind of logical relationships are acceptable between the IO parameters of a service and the IO parameters of a request [4]. However, this kind of matchmaking takes into consideration the whole IO signature and can only calculate the degree of match between one service and one request.

While we agree that logical signature matchmaking is important to check that the IO signature of a service is compatible before using it for a task that requires specific IO paramters, we argue that complex mechanisms such as service composition or service provisioning require a more flexible approach than the rigid matchmaking filters discussed in [4]. We build our logical signature matchmaking method based on the concept of individual *Links* between a source parameter and a destination parameter.

A. Links

We define a *link* as a logical relationship between two IO parameters. A link has a source parameter $Source$ and a destination parameter $Destination$ and is denoted $Link(Source, Destination)$. Links in automated service

[5]http://alias-i.com/lingpipe/

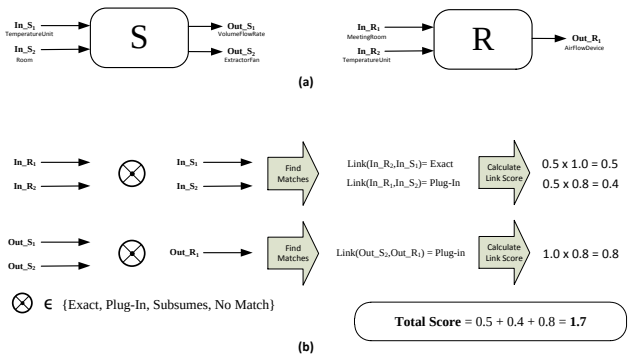

Fig. 3. Link Weight Matching example.

matchmaking can represent a possible connection between two services, the dependency of an input of a service on one of the input parameters specified in a service request, or the ability of a service to generate one of the outputs specified in a service request. The definition of links defined in this paper derives from the definition of *Casual Links* [22].

Given a domain ontology model τ, a casual link between the output parameter Out_s_y of service s_y and the input parameter In_s_x of service s_x can belong to five different categories:

1) **Exact**: if Out_s_y and In_s_x are equivalent concepts; formally, $\tau \models Out\_s_y \equiv In\_s_x$.
2) **PlugIn**: if Out_s_y is a sub-class of In_s_x; formally, $\tau \models Out\_s_y \sqsubseteq In\_s_x$.
3) **Subsumes**: if Out_s_y is a super-class of In_s_x; formally, $\tau \models Out\_s_y \sqsupseteq In\_s_x$.
4) **Intersection**: if the intersection of Out_s_y and In_s_x is satisfiable; formally, $\tau \not\models Out\_s_y \sqcap In\_s_x \sqsubseteq \bot$.
5) **Disjoint**: if Out_s_y and In_s_x are incompatible; formally, $\tau \models Out\_s_y \sqcap In\_s_x \sqsubseteq \bot$.

Our definition of a link differs from casual links in that we specify that every link has a source parameter and a destination parameter and does not always necessarily exist only from an output of a service to the input of another service. While casual links are only applied in service composition where the output of one service is linked to the input of another service, our definition of a link can also be used to perform logical signature matchmaking between a service and a request. Checking individual links makes it possible to assess the degree of match between a service and a request more flexibly than with rigid logic filters such as those described in [4].

We argue that a *Subsumes* link between an output Out_s_y of service s_y and the input In_s_x of service s_x cannot be used in practical cases because the super-class of a parameter is more general and may consist of other sub-classes of parameters which In_s_x is not compatible with and would result in service s_x not being able to work properly. The same argument applies for *Intersection* links. The only instance in which a *Subsumes* link is applicable is when the output of a service is linked to an output of a request. In such a case, if an *Exact* or *PlugIn* link does exist, providing a super-class of the desired output parameter as the final output is better than not providing any output at all. Thus in our work, a link can belong to only one of the four categories defined below.

Let τ be a domain ontology model. Let $Source$ be a source IO parameter concept and let $Destination$ be an IO parameter concept that $Source$ can be linked to. Then, the type of link between $Source$ and $Destination$: $Link(Source, Destination)$ can be classed as one of the four categories explained below:

1) **Exact**: $Source$ is an exact match to $Destination$ if $\tau \models Source \equiv Destination$.
2) **PlugIn**: $Source$ plugs into $Destination$ if $\tau \models Source \sqsubseteq Destination$.
3) **Subsumes**: $Source$ subsumes $Destination$ if $\tau \models Source \sqsupseteq Destination$.
4) **Disjoint**: $Source$ is not related to $Destination$ in any of the above ways.

Note that although we didn't drop the *Subsumes* link, we only allow such links when linking the output of a service to the output of a request.

B. Weighted-Link Matchmaking

We propose *Weighted-Link Matchmaking* as a means to measure the logical signature match between a service S and a request R. A weighted-link match operates separately on each one of the IO parameters making the logical signature of a service. For matching the inputs of a request to the inputs of a service (an input-input link), the total link score that can be assigned to a link $T_{w_{in}}$ depends on the number of inputs of the service *i.e.*

$$T_{w_{in}} = \frac{1}{|in(S)|} \tag{3}$$

For matching the outputs of a service to the outputs of a request (an output-output), the total link score that can be assigned to a link $T_{w_{out}}$ depends on the number of outputs specified in the request *i.e.*

$$T_{w_{out}} = \frac{1}{|out(R)|} \tag{4}$$

The maximum weight given to an input-input link depends on the number of inputs of the service rather than the inputs of the request because the highest priority here is to make sure that all the inputs necessary for the service to operate can be satisfied. If one of the inputs is missing, the service cannot be used properly while it is ok to leave one of the inputs specified by the request unused. Conversely the maximum weigth given to an output-output link depends on the number of outputs specified in the request. The reason behind this is that the imporant aspect is whether a service can generate all the outputs required by the request. In automated systems, it could be acceptable that a service generates an extra output if that output is not used. What matters is that all the outputs

specified in the request are ultimately generated and supplied to the service consumer.

For example, service S shown in Figure 3a. can provide two outputs but only one output parameter is specified in the request R. Thus $T_{w_{out}} = 1$.

We define a weight function w_f that assigns a weight to the strength of a link between a source parameter S_{rc} and a destination parameter D_{st} depending on the type of the link.

$$w_f(Link(S_{rc}, D_{st})) = \begin{cases} 1.0, & if Link(S_{rc}, D_{st}) = Exact \\ \alpha, & if Link(S_{rc}, D_{st}) = PlugIn \\ \beta, & if Link(S_{rc}, D_{st}) = Subsumes \\ 0.0, & if Link(S_{rc}, D_{st}) = Disjoint \end{cases} \tag{5}$$

where α and β are penalizing weights that allow the user to bias the algorithm towards preferred link types.

For example, from Figure 3b., if we select $\alpha = 0.8$ the degree of match between input parameter In_R_2 of the request and input parameter In_S_1 of the service is $Link(In\_R_2, In\_S_1) = Exact$, thus $w_f(Link(In\_R_2, In\_S_1)) = 1$.

The weighted-link score is calculated using the equation:

$$Link_{Score}(S_{rc}, D_{st}) = T_{w_x} w_f(Link(S_{rc}, D_{st})) \tag{6}$$

where $x \in \{in, out\}$ depending on whether the link is an input-input link or an output-output link. The total matching score $Match_{Logic}(S, R)$ between service S and request R is given by adding the weighted-link score of all the links between S and R:

$$Match_{Logic}(S, R) = \sum_{IO} Link_{Score}(S_{rc}, D_{st}) \tag{7}$$

For example, in Figure 3b, the total logical signature match between service S and request R is $0.5 + 0.4 + 0.8 = 1.7$.

V. Hybrid Semantic Matchmaker

In this section, we explain the hybrid matchmaking approach proposed in this paper. The matchmaking relies on the probabilistic matchmaking component described in Section III to find a short list of candidate IoT services which is then passed to the weighted-link matchmaking component described in Section IV-B to accurately arrange the results.

The probabilistic component is first used to match IoT services to the request based on latent factors extracted from the underlying concepts in the IoT service descriptions. This approach helps to identify statistical similarity between a service and a request and can find relevant candidate services that would otherwise have been ommitted by strict logic matchmaking [16]. The probabilistic component then passes a short list of results to the logic-based component, thus restricting the scope of search for this component and reducing the complexity of the matchmaking. The size of the short list is specified by the user depending on the number of service required.

The logic-based compoment verifies the IO signature of each candidate service and calculates the weighted-link score. Finally, the results from the logic based are ranked based on their weighted-link score. In case of a tie, the score from the probabilistic compoment is used as a tie-breaker. The final ranked list of results is presented to the client.

VI. Evaluation

Many of the existing works on semantic service matchmaking are based on the OWL-S model [4], [5]. In order to compare our approach with state-of-the-art service matchmakers, we perform the comparative analysis in this paper using the OWL-S service retrieval test collection OWLS-TC v3.0[6]. The close relationship between the OWL-S service model and the IoT service modelling framework was explained in Section II. The dataset consists of 1007 service descriptions defined in OWL-S form. The services are divided into seven categories and a total of 29 OWL-S queries are provided together with a relevant answer set for each query. The answer set for each query consists of a list of relevant service and each service i has a graded relevance value $label(i) \in \{1, 2, 3\}$ where 3 denotes a high-relevance to the query and 1 denotes a low-relevance. Table I shows the number of services and queries belonging to each of the seven categories.

TABLE I
Number of Services and Queries for each domain.

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

The probabilistic method (based on LDA) described in Section III and the hybrid method described in Section V are compared with a text-matching approach powered by Apache Lucene[7] and also methods from the OLWS-MX 2.0[8] hybrid semantic Web service matchmaker (M0, M3, and M4) [5]. M0 is a logic-based approach and M3 and M4 are hybrid approaches which use both logic and non-logic based methods. In the next section, the probabilistic method based on LDA is labeled *LDA* and the hybrid method is labeled *LDA + Logic*.

For the hybrid method, in these experiments we have given a higher weight to PlugIn links and have penalized Subsumes links. The parameters α and β are set to 0.8 and 0.4 respectively based on heuristic measures. The size of the short list which should be specified by the user was set to 40 services since our evaluations were carried out to up to 40 services retrieved.

The sample queries are all in the form of OWL-S templates and contain the semantic requirements together with a text

[6]http://www.semwebcentral.org/projects/owls-tc/
[7]http://lucene.apache.org/
[8]http://semwebcentral.org/projects/owls-mx/

description of the queried functionality. For the text-based approach, the text descriptions of the service attributes are retrieved from the query templates and used as the query string.

We evaluated our approach by calculating the *Precision at n* ($P@n$) and the *Normalised Discounted Cumulative Gain* ($NDCG_n$) for the results obtained for each of the sample queries. These are standard evaluation techniques used in *Information Retrieval* to measure the accuracy of a search mechanism with respect to completeness of the results returned.

1) Precision @ n: Precision is a measure used to evaluate the results of the search and matchmaking process. Precision @ n 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:

$$precision = \frac{|\{RelevantServices\} \cap \{RetrievedServices\}|}{|\{RetrievedServices\}|} \tag{8}$$

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

2) Normalised Discounted Cumulative Gain: $NDCG_n$ is a measure that takes into account the graded relevance of each service retrieved. This measure is particularly useful for evaluating ranking results since not all services in a relevance set are of the same relevance to the query. The $NDCG_n$ for n retrieved services is given by Equation 9.

$$NDCG_n = \frac{DCG_n}{IDCG_n} \tag{9}$$

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 query. The DCG_n is calculated by Equation 10.

$$DCG_n = \sum_{i=1}^{n} \frac{2^{label(i)} - 1}{log_b(1+i)} \tag{10}$$

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: $b = 2$; persistant: $b = 14$).

$NDCG_n$ gives higher scores to systems which rank services with higher relevance first and penalizes systems 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).

VII. Results

The average $P@n$ and $NDCG_n$ are obtained over all 29 queries for LDA + Logic, LDA, Pure Text-Matching, OWLS-M0, OWLS-M3, and OWLS-M4. The results are shown in Figures 4 and 5 respectively. The Precision@n results show that Pure 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. LDA used the information captured in the latent factors to match services based on statistical similarity rather than just semantic or syntactic similarity and thus exhibited better precision than Text Matching and OWLS-M0. OWLS-M3 and OWLS-M4 also found more relevant services than the Pure Text-Matching and the logic-based OWLS-M0. LDA performed better than OWLS-M4 but OWLS-M3 exhibited better precision than LDA for the first 15 services retrieved. The hybrid LDA + Logic method successfully combined the merits of LDA with weighted-link matching to accurately rearrange the results thus outperforming all the other methods in terms of precision.

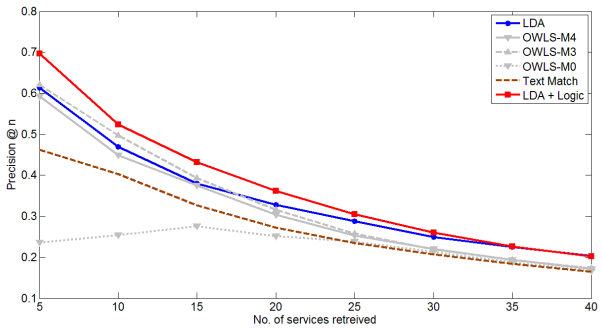

Fig. 4. Comparison of average $P@n$ values over 29 queries

$NDCG_n$ evaluates the ranking mechanism and it is the most important measure for automated search and matchmaking engines. The top most relevant (*i.e.* the first five or ten) results retrieved by a search and matchmaking engine are the main results that will be used by the client. The LDA and LDA + Logic matchmakers perform better than the other search and matchmaking mechanisms in this experiment. LDA + Logic holds a higher $NDCG_n$ than all other methods for any number of services retrieved, this reflects the accuracy of the hybrid ranking mechanism used by our method. Pure 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. OWLS-M3 and OWLS-M4 both exhibit a high $NDCG_n$ but they are outperformed by the LDA and LDA + Logic matchmakers.

VIII. Conclusions

Web services provide an ideal solution to enable machine-controlled and automatically structured service-oriented dynamic systems in Internet of Things. Semantic service matchmaking is the fundamental construct on which higher level service-oriented functionalities such as IoT service recommendation, composition, and provisioning are provided.

The hybrid semantic matchmaker for IoT Services proposed in this paper combines probabilistic matchmaking with a

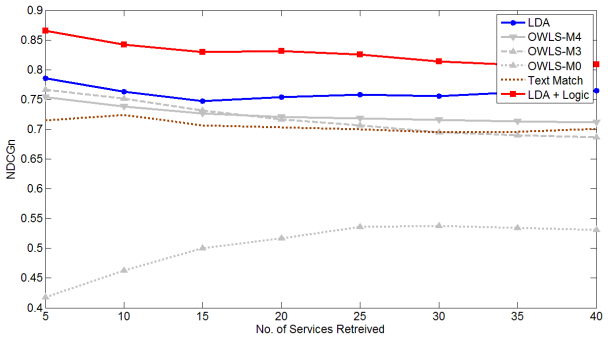

Fig. 5. Comparison of average $NDCG_n$ values over 29 queries

logical signature matchmaking method. The probabilistic component uses a latent semantic analysis model to extract latent factors from the IoT Service description data and uses these latent factors to overcome problems often encountered with logic-based techniques such as semantic synonomy. However, probablistic service matchmaking alone does not check the IO signature of a service. Therefore, logic-based IO signature matchmaking is important when specific input and output parameters are needed such as in service composition or service provisioning scenarios.

The proposed method exhibits higher performance than existing methods in terms of $P@n$ and $NDCG_n$. The weighted-link matchmaking provides a versatile approach for evaluating the degree of match of individual links and paves the way for the integration of the hybrid semantic service matchmaking method with higher-level service-oriented functionalities in the Internet-of-Things. Future work will focus on creating an automated IoT Service composition solution that uses our hybrid semantic matchmaker to find candidate services for composition and/or compensation in the Internet-of-Things.

ACKNOWLEDGMENT

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.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Comput. Netw.*, vol. 54, pp. 2787–2805, October 2010.

[2] F. Chen, C. Ren, J. Dong, Q. Wang, J. Li, and B. Shao, "A comprehensive device collaboration model for integrating devices with web services under internet of things," in *Web Services (ICWS), 2011 IEEE International Conference on*, july 2011, pp. 742 –743.

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] G. Cassar, P. Barnaghi, and K. Moessner, "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.

[17] S. De, P. Barnaghi, M. Bauer, and S. Meissner, "Service modelling for the internet of things," in *Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on*, sept. 2011, pp. 949 –955.

[18] M. Compton and et al, "The ssn ontology of the w3c semantic sensor network incubator group," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 0, no. 0, 2012.

[19] J. Kopecký, K. Gomadam, and T. Vitvar, "hrests: An html microformat for describing restful web services," in *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, ser. WI-IAT '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 619–625. [Online]. Available: http://dx.doi.org/10.1109/WIIAT.2008.379

[20] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003.

[21] M. Steyvers and T. Griffiths, *Latent Semantic Analysis: A Road to Meaning*. Laurence Erlbaum, 2007, ch. Probabilistic topic models.

[22] F. Lécué, E. M. Goncalves da Silva, and L. Ferreira Pires, "A framework for dynamic web services composition," in *2nd ECOWS Workshop on Emerging Web Services Technology (WEWST07), Halle, Germany*. Germany: CEUR Workshop Proceedings, November 2007.