Learning Mechanisms on OWL-S Service Descriptions for Automated Action Selection

Johannes Fähndrich, Nils Masuch, Lars Borchert, and Sahin Albayrak

DAI-Laboratory of the Technische Universität Berlin
Department of Electrical Engineering and Computer Science
Ernst-Reuter-Platz 7, 10587 Berlin, Germany
johannes.faehndrich@dai-labor.de (Corresponding author)

Abstract. With the increase complexity of IoT systems, the well known development paradigm for software development like Service Oriented Architectures and Multi Agent Systems have to be adapted to fit the now challenges of IoT environments. Here the vast amount of available components and the specific implementation for special purpose hardware calls for an abstraction of functionality as well as to new development tools which allow to handle the dynamism of IoT applications. Here a multitude of special purpose sensors with view computational resources have to be combined to create emerging software. Most of the hardware reaches an return of investment only offering its service to third party developers. For a developer to be able to integrate those services into an application, a adaptive search mechanism need to be developed, which is able to specialize in the different domains of e.g. IoT applications.

1 Introduction

Distributed systems based on the Service Oriented Architecture (SOA) paradigm have become more and more popular in recent years due to different trends like the Internet-of-Things or Big Data. This approach is accompanied by the planning community that chose planning based upon services as one subject to research. However, there are some challenges that come up with this vision. Usually services are not described by one central entity but by different humans, which—in their domain—have a context dependent view on the described objects of the world. A first step to cope with this challenge is the usage of one domain representation that all service developers use for their respective service description. Nevertheless, the way how each developer defines relevant attributes like input, output, preconditions or effects can be extremely different, which makes the general interpretation of a service's functionality difficult. State-of-the-art approaches therefore integrate service matchmaking components based upon semantic service descriptions, which can be used to find fitting services in each state of an agent planning algorithm. Therefore service matchmaking can be interpreted as one step planning and can seamlessly be integrated into the action selection mechanism of an AI planning algorithm in order to find a connection between the initial state and the goal state within the search space. This approach has been subject to research and will be surveyed in the next section (see Section 2).

Usually service matchmaking components do not rely on one single matching strategy but use different mechanisms to compare a service request with an advertisement. One question that arises out of that is how to weight each strategy in order to aggregate them to a single result. The optimal weighting can depend on several attributes like the application domain, the quality of the ontology, the completeness of information of the service descriptions and the language being used for the description. Due to this volatile optimal weighting we propose to integrate learning mechanisms into the service matchmaking process, which adapts the weightings to the problem dynamically.

In a prior work [8], we outlined an iterative process that uses a service matchmaker to find matching services in the action selection process of a planning process. In this paper we describe the implementation and evaluation of the new matching aggregation process and different learning approaches for defining an optimal weighting for the semantic service matchmaker SeMa^2.

The remainder of the paper is structured as follows. In Section 2 we will present and discuss the related work. This includes an overview of service description languages, service matchmakers and service planners. In Section 3 we introduce the challenges associated with automated service planning and describe our concept of extending $SeMa^2$ by a comprehensive aggregation process. Subsequently, in Section 4 our approach of learning the weights for different matching techniques is presented in detail and evaluated. Eventually, we conclude the work and give an outlook on future work in Section 5.

2 Related Work

In this section we will at first give an overview about the related work of semantic service matchmakers followed by learning approaches that make service matchmaking adaptable.

2.1 Service Matchmaker

M. Klusch and *P. Kapahnke* [15] present a hybrid service matchmaking component called *iSeM*, which performs logical as well as syntactical and structural matching. iSem participated in the 2012 S3 Contest and was the only matchmaker able to process the provided PE service specifications in SWRL. Besides logical matching filters for input and output parameters the solution applied a strict logical specification plug-in matching. This means, that the component checks whether there exists a transformation of the requested/provided preconditions/effects, in order to infer from a requested precondition to a provided one and from a provided effect to a requested one. This process is called θ-subsumption and is in the case of iSem provided without any consideration of instances. The inferencing is being done after SWRL rules are converted into PROLOG.

Another work performing PE matching is SPARQLent [19], which not only performs matchmaking but also planning. It participated in the 2012 S3 OWL-S Contest and assumes that PE are described in the query language SPARQL. Hence, the selection process is based on query containment relations not only considering PE, but also input and output concepts.

In contrast to the other approaches the work of *Valle et al.* [4] named *GLUE* is based on WSML. Since WSML already comes with a conceptual model of service discovery, this work proposes a refinement that has a specific focus on mediating goals of different ontologies. The reasoning process itself is performed with F-Logic.

In the work of *Lamparter et.al.* [16] the authors present a proprietary service definition based on OWL-DL. Beside input and output parameters, the algorithm also considers pricing functions described in SWRL and configurations under which a service is executable. The latter refers to some form of condition checking, since the requester can search for services that provide a specific, desired configuration. The reasoning on services is being done with SPARQL queries.

Bener et al. [2] extend SAM (Semantic Advanced Matchmaker) by PE matching strategies based on OWL-S and SWRL. The matching procedure for conditions is separated into four matching modules, namely subsumption based scoring, semantic distance scoring, WordNet based scoring, finalized and aggregated via a bipartite matching approach. The work introduces weights for the aggregation of different matching results, which are shown in Table 1 taken from [2].

Table 1. Scoring assessments

Relationship	**Score**
Exact	1.0
Plug-in for Preconditions	0.6
Plug-in for Effects	0.4
Subsume for Preconditions	0.4
Subsume for Effects	0.6
Fail	0.0

The weights in this case are fixed and thus not learned. Furthermore, those weights only concern discrete level of matches.

In conclusion the described approaches rely on different languages for the description of conditions ranging from decidable ones, such as OWL-DL and WSML-DL to undecidable ones, such as SWRL, F-Logic and PROLOG. However, in most of the related work on service matchmakers regarding PE matching the undecidable rule languages have been limited in its expressiveness in order to guarantee termination. So far, the service matchmakers have included different similarity measures and have introduced some weights to model their importance against each other. The weights started out to classify two types of similarity measures and got more detailed to the point where different parts of the service description like preconditions and effect are weighted differently. These approaches have one fact in common: the choice of the weights are fixed and do not adapt to the context of use.

2.2 Learning Service Matchmaker

In order to optimize the result of the service matching a learning phase can be introduced to adjust the parameters of a service matchmaker to the properties of the domain. The parameters to learn depend on the service matchmaker and thus its flexibility depends on the parameters that can be observed.

In *Klusch et al.* [14] the authors introduced a formal model of defining weights for the aggregation of different similarity measures with the names w_w-similarity and w_s-structural similarity measure. The aggregation method has been learned using a Support Vector Machine (SVM) approach based on training data. The matchmaker component that invokes this approach is designed to match SA-WSDL services.

M. Klusch and *P. Kapahnke* [13] introduce another learning service matchmaker by extending the approach of a prior work [11] for OWL-S service descriptions. Here matching results of different matching types are aggregated using a weighted mean. The authors introduce different types of matching results that are weighted. Firstly, approximated logical matching, which is divided into approximated logical plug-in and subsumed-by matching. Secondly, non-logic-based approximated matching, which are text and structural semantic similarity-based signature matching. The weights of this aggregation are also learned using a SVM. This supervised learning approach is replicated in our work, but with a different learning algorithm. The relevance set that is used to rank the matching results are reused with a genetic algorithm and a hill-climbing search.

To the best of our knowledge there exist only these both approaches that utilize machine-learning techniques in order to cope with the challenge of aggregating service matchmaking techniques.

3 SeMa^2 and its Expert System

The service matchmaker SeMa^2 [17] follows a hybrid approach combining logic-based and non-logic-based matching techniques using OWL-S and SWRL as part of a single-agent system. In a subsequent work [8], an extension of SeMa^2 was proposed, where each of the matching techniques (in the remainder named similarity measures) are encapsulated in one agent. The resulting multi-agent system presents a group of experts, each able to express its opinion about a match by means of a probability. In the following, we will describe the implementation of this extension with a selected set of similarity measures. Afterwards we will present the probabilistic model used to aggregate the different expert opinions.

3.1 The Expert System

In this section we present the extended architecture of SeMa^2 and discuss the relation between different matching experts. An overview of the expert system is illustrated in Fig. 1.

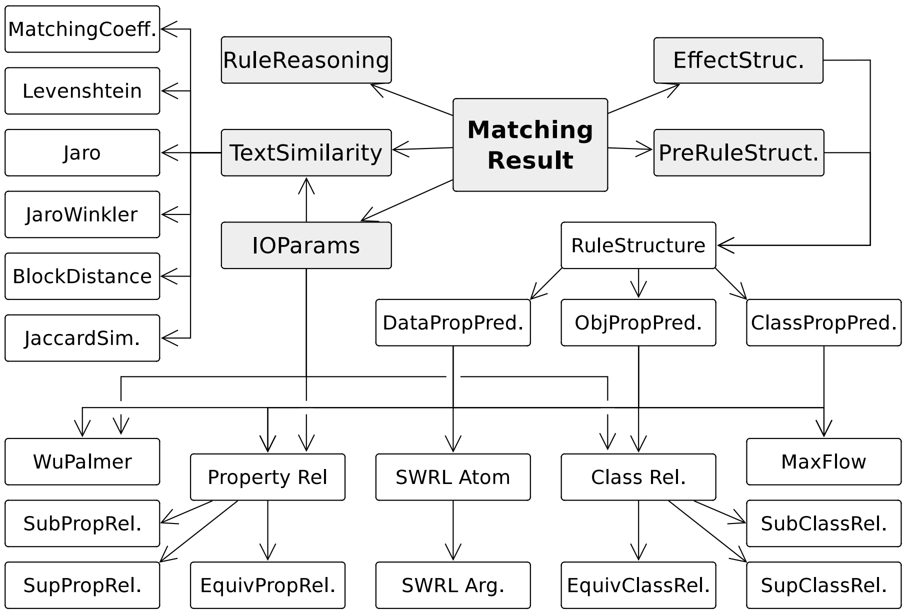

Fig. 1. Expert structure of the SeMa^2 architecture.

The schematic depiction shown in Fig. 1 highlights that lower experts are used by the upper ones to build opinions. Each node in the figure is an expert type. Each expert can have multiple instances in the implementation depending on its use in the hierarchy. The edges of the tree represent weights, which are used to aggregate the opinion of the lower expert into the upper experts opinion. For example, the *TextSimilarity Expert* uses six other experts to establish its opinion. In total, the SeMa^2 consists of 81 experts. We will now have a look at different excerpts of the experts.

We start with the overall result of the matching process, which is the opinion of the *MatchingResult Expert* which aggregates the matching results of the *TextSimilarityExpert*, the *PreRuleStructureMatcherExpert* (structural precondition matching), the *EffectStructureMatcherExpert* (structural effect matching), the *IOParamMatcherExpert* (structural input/output matching) and the *RuleReasoningMatcherExpert*, responsible for validity checks of instances on the service's rules. Fig. 1 shows those experts in gray.

The comparison of the arguments of a rule's predicate presents the next improvement opportunity. Here the used algorithm abstracts from the order the parameters are used but uses a 'is equal' as comparison. This has the advantage to produce a binary acceptance criteria but lacks the ability to recognize hypernym or hyponym concepts. The proposed extension here is the use of a probabilistic measure which is able to distinguish a semantic distance between the concepts of interest. Such measures are subject of research, e.g. *Benner* [2] uses such measures for service matchmaking. Again, we define such measures as expert opinion and use an information fusion method to aggregate such opinions.

3.2 Scoring and aggregation of matching techniques

There are many different similarity measures to extend the PE matching in a probabilistic framework similar to the related work (cf. [2, 15]). Some of them are: *Semantic distance based scoring* [22] analyzing the embedded ontology of two concepts to find the shortest path from one concept to another; *WordNet based scoring* [2] that can be used to find lexical similarity in used words. If two concepts out of different ontologies need to be matched a *bipartite matching score* is able to rate the similarity by i.e. the maximum cardinality match counting the edges between the different concepts. More sophisticated methods use ontology matching to find a semantic relation between concepts. Logic based scoring like proposed in *Approximated Logical Matching* [15] can be interpreted as similarity measures using reasoning on formal features of the rules describing the preconditions and effects.

Those similarity measure each compute a similarity score interpreted as an probability where 1.0 is a perfect match and 0.0 is no match at all. We now have a look on how a probabilistic model can formalize such an expert opinion.

Probabilistic model of opinion Like in our previous work [8] we apply the results of *Morris* [18] to model the expert opinions as probabilities $p_i(R, S)$. We will have a closer look into this formalization in the following section. As an expert observes two concepts and elaborates their semantic distance we can abstract its opinion as $p_i(\Theta|d)$ where Θ is the subject of interest and d are the observations. An expert can then collect evidence for its opinion by conducting multiple observations d_i. Each observation might then be interpreted as evidence to strengthen the experts' opinion. Following *Beyerer* [3] a Bayesian interpretation of the conditional probability $p_i(\Theta|d)$ could be interpreted as a degree of confidence or even better as a degree of belief. With such an interpretation we can use this formalism to model the expert opinions as described in the following equation:

$$\underbrace{p(\Theta|d)}_{A-Posteriori} = \frac{p(d|\Theta)p(\Theta)}{p(d)} \propto \overbrace{p(d|\Theta)}^{Likelihood-Function} \underbrace{p(\Theta)}_{A-Priori} . \tag{1}$$

Here the subject of interest is Θ, *i.e.* the equivalence of a Horn-clause. The observations or information used by the expert to assess its opinion is formalized in d. An example of this d could be the attached ontologies to the concept in order to calculate the semantic distance. The expert can update its opinion after observing another d using Bayesian fusion by calculating the product described in Eq. 1. If, for example, one concept is a hypernym of the other, $p(d|\Theta)$ could be proportional to the minimal distance between those two concepts [22]. Furthermore, $p(\Theta)$ allows the expert to formalize a-priory knowledge about the probability of Θ.

For the same request and advertisement pair the different measures might have different similarities. And with that, the experts have different opinions. If this happens, an aggregation of the different opinions needs to be found.

Opinion aggregation As we have shown elsewhere [8] different aggregation functions -so called pooling methods- can be used to aggregate the opinions of experts. The opinions $p_i(\Theta|d)$ are collected and need to be fused to one score. Since the experts are not always equally important the possibility to prioritize by weightings of the different expert opinions is a requirement for the fusion method. With the probabilistic formalization of the expert opinions method like the *Dempster-Schafer theory of evidence* [20], fuzzy logic or artificial neuronal networks can be used for information fusion [6]. Subsequently to prior work [8] this work uses a method of opinion aggregation called pooling method formalized in a function $K(p_1, \dots, p_n)(\Theta)$. It is acquired by adapting a weighted mean to the aggregation of opinions. We choose a weighted arithmetic mean called linear opinion pool [21] as our first pooling method. This arithmetic mean has been generalized by *Genest* [9] to be able to use weights in the interval $[-1, 1]$ in a more general class of linear opinion pools (GenLinOP). This opinion pool has the form of Eq. 2.

$$K(p_1, \dots, p_n)(\Theta) = \sum_{i=1}^{n} w_i p_i(\Theta) + \left[1 - \sum_{i=1}^{n} w_i\right] R(\Theta) \tag{2}$$

Here, $w_1, \dots, w_n \in [-1, 1]$ are weights representing the confidence in an expert and R is an arbitrary probability function, with the restriction:

$$\forall J \subseteq \{1, \dots, n\} : \left|\sum_{j \in J} w_j\right| \leq 1.$$

The method shown in Eq. 2 has been chosen because of its theoretical sound standing. Other pooling methods have been and are continued to be evaluated which is subject to research. The GenLinOP has the possibility to include—besides the opinion of the group—an a-priori established probability which can be modeled as $R(\Theta)$. We normalize the weights to $\left|\sum_{j \in J} w_j\right| = 1$ which lets us neglect the a-posteriori opinion $R(\Theta)$.

The GenLinOP is compared against the Logarithmic opinion pool [10] (LogOP) and the weighted harmonic mean (HARM) since the GenLinOP has several theoretic weaknesses [1] like allowing dictatorships. There is a infinite number of pooling methods where each of which has its own properties which are of concern in different applications. The LogOP for example has the curios behavior of a 'veto'. Meaning that if a single expert has the opinion of $p_i(\theta) = 0$ then the aggregated opinion is $K(p_1(\theta), ..., p_n(\theta)) = 0$ independent of the given weights. Eq. 3 describes the LogOP:

$$K(p_1(\theta), ..., p_n(\theta)) = \frac{\prod_{i=1}^{n} [p_i(\theta)]^{w_i}}{\sum_{\theta \in \Theta} \prod_{i=1}^{n} [p_i(\theta)]^{w_i}} \tag{3}$$

with weights $w_i \in [0, 1]$ and $\sum_{i=1}^{n} w_i = 1$.

One of the properties which make the LogOP interesting for the opinion aggregation is that it as what is called the unanimity property which states that if all experts have the same opinion, the aggregated opinion is equal to this opinion.

The third aggregation method we will analyze is the weighted harmonic mean.

$$K(p_1(\theta), ..., p_n(\theta)) = \frac{\sum_{i=1}^{n} w_i}{\sum_{j=1}^{n} \frac{w_j}{p_j(\theta)}} \Bigg| \, w_i \in [0, 1]; \sum_{i=1}^{n} w_i = 1; p_j(\theta) \neq 0. \tag{4}$$

The harmonic mean has the down side of not allowing opinions of 0. Which leads to the interpretation of the expert opinion that there is no total miss match at all. Such an postulate is subject to discussion and depends on the domain, which is model by the experts.

Taking this theoretical framework as a basis, we implement the different measures used in the service matching as experts returning a probability $p_i(\Theta)$ and aggregate them with a pooling method $K(p_1, \ldots, p_n)(\Theta)$. For an example we have adapted the comparison of the arguments of a predicate. The expert opinion modeled as probability is as follows:

$$p(\Theta) = \begin{cases} dist(a_r, a_s)^{-1} & \text{, if } 1 \geq dist(a_r, a_s) > 0 \\ 1.0 & \text{, if } a_r.getIRI() \equiv a_s.getIRI() \\ 0 & \text{, else} \end{cases} . \tag{5}$$

This leads us to the question on how to measure the distance between concepts. This question has been subject to research, e.g. by *Euzenat* [5]. The next section gives a short overview of one conceptual metric implemented in our approach as proof of concept.

In Eq. 5 the first case ($dist(a_r, a_s)^{-1}$) defines the distance between the two concepts in different ways regarding the distance appropriate to the properties of the concept. Each distance measure is implemented in one expert and we have e.g. identified Class-Property, Data-Property, Object-Property and Class-Relationship experts in SeMa[2]. Class properties are well know and often analyzed in the related work. The Pellet reasoner supports reasoning on class properties and as example distance between concepts the Wu-Palmer measure [23] shown in Eq. 6 has been implemented in one expert.

$$dist_{WP}(a_r, a_s) = \frac{2 \times \eth(\triangle(a_r, a_s))}{\eth(a_r) + \eth(a_s)}. \tag{6}$$

With $\eth$ being the distance to the root element of the tree and $\triangle$ is the lowers common subsumer of the concepts a_r and a_s. Since the for Class-Properties $dist_{WP}$ can be calculated using Pellet but for Object- and Data-Properties Pellet denies reasoning support so an own implementation has been created.

In a similar manner all other similarity measures are turned into probabilistic expert opinions. With this change e.g. we are able to distinguish partial argument matches. We want to emphasize the importance of such a partial match for planning tasks. Here multiple services can be used to fulfill the arguments of a predicate in a precondition. Thus on a higher level, we are able to use multiple services to fulfill the preconditions of a successor task or service.

The matching on arguments is done by a bipartite graph matching solved with the max flow algorithm, where the edges between request and advertisement nodes are the opinion of the expert. Fig. 2 shows an example of such an bipartite graph matching problem.

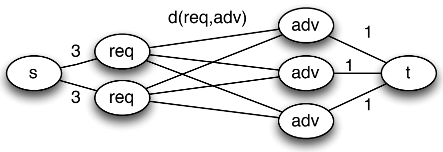

Fig. 2. Example of the max flow problem to solve for a bipartite graph matching.

Here the edges exiting the source s have the weight of the sum of the edges entering the sink t because in this way if one request parameter matchmaker all advertisement parameters the maximal flow can be seen as perfect match. The result of the maximal flow though this graph is divided by the sum of the weights of edges which enter the sink and forms the opinion of the 'MaxFlowExpert'.

One particular challenge is the measure for a class relationship in regards to a matching tasks. In the related work this is sometimes called subsumes and subsumed-by relation (cf. [15]). The question at hand here is: If an exact match of concepts in a ontology is detonated with 1.0 and no match is 0.0 how much is a sub- or super match? Since all choices done here would be arbitrary, we decided to let the service matchmaker learn those parameters in an offline stage prior the actual matching. The grey marked sets (relationship experts) in Fig. 1 are such experts, where only one expert might return 1.0 and the other converge to 0.0. In this case the weights for these experts denote the influence of an exact-, super- or sub-match.

The learning is done via a genetic algorithm using the Watchmaker Framework[1]. Where the weights for the 81 experts is seen as a DNA pattern and are mutated at each generation. The Mutation and the details of the assessment of each generation is explained in the next section.

The result of the learning is a weight vector describing the confidence in an expert regarding its opinion, which are used in the Pooling Methods. Since the expert structure is not flat some of the experts depend on opinions of other experts and with that the weights are dependent as well. Furthermore, the learning takes place on the expert instances and not on a general scope since the same expert, e.g. the TextSimilarityExpert, can be used to evaluate different properties of a service description. For example, the name of the service might have more affect on the service matching result then a text similarity of some parameter. Thus the weights for those each instances has to be learned separately.

Further it is questionable if the learning of the weights of the pooling method is significant.

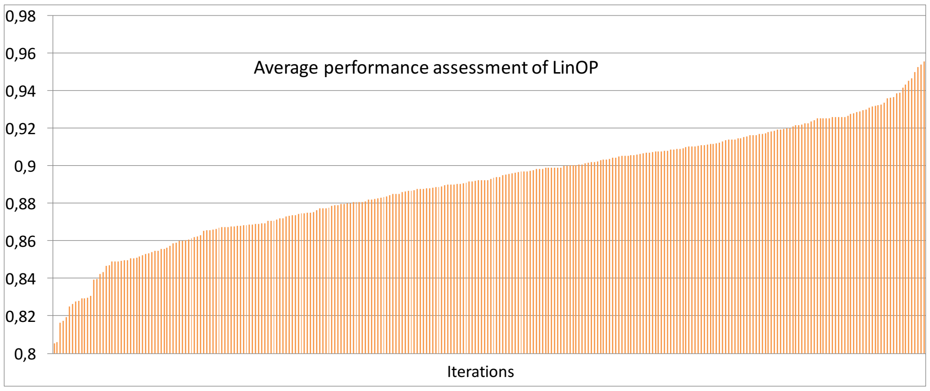

Fig. 3. Example weight generations during the learning (X-axis) and their assessment (Y-axis) of the LinOP.

As shown in figure 3 the different weight vectors yield different results of the LinOP reaching form 0.8 to 0.95 thus the selection of weights is significant.

4 Learning the Weights

This section will introduce the learning approach for the weights of the aggregation methods and presents the first experimental results. As mentioned above we used genetic algorithms to optimize the weights of the aggregation methods.

Similar to *Klusch* [12] the learning is done in an offline phase before the matching task. Giving the service matchmaker an opportunity to adapt to the domain it is used in.

[1] http://watchmaker.uncommons.org/

Learning algorithm We used the Semantic Web Service Matchmaker Evaluation Environment (SME) [2] for the evaluation. The test collection of services and request from the S3 Contest[3] with 1080 services and 42 requests has been used. This test collection provides a relevance rating consisting of sets of relevant and irrelevant services for the 42 requests. This is used to evaluate the matching performance of the service matchmaker with varying weights. As quality measure the normalized discounted cumulative gain (NDCG) is used. This quality measure optimizes the amount of relevant service with high matching values as shown in Eq 7.

$$DCG = \sum_{i=1}^{n} \frac{2^{rel_i} - 1}{log_1(i+1)} \Bigg| NDCG = \frac{DCG}{Best\ DCG} \tag{7}$$

With these preconditions a supervised learning method has been chosen: Genetic Algorithms. We used the Watchmaker Framework for the support of the generic algorithm life cycle. The genome is modeled as a list of all 81 weights each reaching from 0 to 1 with the restrictions among siblings having their weights normalized to sum up to 1. The first generation is initialized with random weights. With each generation the experts with better NDCG values have thus higher weights, until one round of an N-Fold cross validation is completed. We have implemented an mutation function, which changes the weight of an expert depending on the NDCG value he reached. Following this approach the experts with higher NDCG value have bigger weights in the next generation of the weight vector. The evaluation has been done with an N-fold cross-validation with different N reaching from 1 (single fold) over 7 to 42 (leave one out). The results which are shown in the next section have been taken from the 7-fold-cross-validation.

Evaluation In this section we unveil the results of the learning phase of the $SeMa^2$ regarding the different pooling methods. The results have been produced in three experiments where each experiment has run for approximately one day. We have to notice that not the absolute height of the weights are subject to research, since those weights might change with the domain they are learned in. We rather emphasize that the learning yield different results depending on the pooling method and further that the quality of the matching result highly depends on the weights chosen.

The result of the learning process using the HARM as aggregation method are depicted in Fig. 4: One is the average weights (blue) and the other one is the weight configuration yielding the best matching results (orange) of an NDCG 0.9195.

To compare our results to the ones of *Benner* [2] and the ones published by *Klusch* and *Kapahnke* [13] which do weight the first level of similarity measures, we have highlighted the first level of experts in the weight distributions in Fig. 4,

[2] http://projects.semwebcentral.org/projects/sme2/
[3] http://www-ags.dfki.uni-sb.de/ klusch/s3/

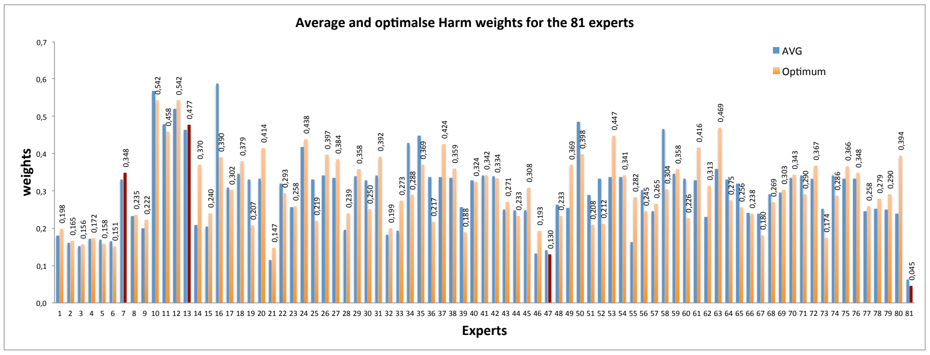

Fig. 4. Learned weights for the 81 experts using HARM.

5 and 6 in the order from left to right: Expert 7 being the *TextSimilarity Expert*, 13 the *IOParamMatcher Expert*, 47 the *PreRuleStructureMatcher Expert* and 81 the *EffectStructureMatcher Expert*. It has to be noted that the hierarchy of experts has multiple levels and only the first one is emphasized here. Additionally the weights are not shown as final weights but rather as the assigned weights. To get the actual influence of the expert to the overall result the weights in the hierarchy need to be multiplied so that the relevance of the children is weighted with the relevance of the parent expert. This is not done here since the overall weights are not of interest since they are domain dependent.

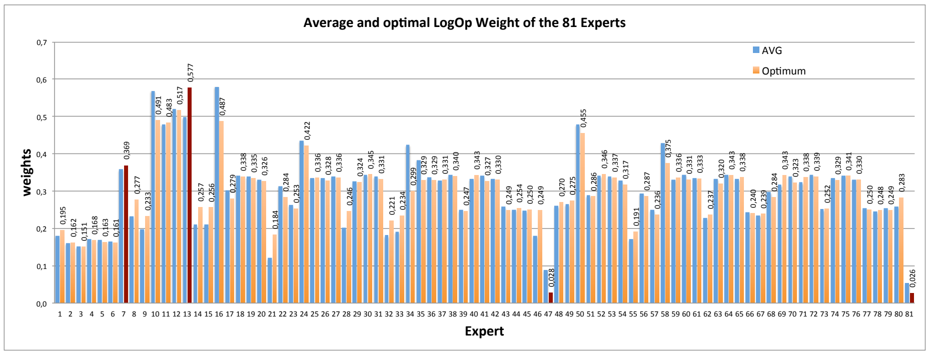

Fig. 5. Weighted logarithmic Mean

Secondly the resulting weights of the learning using the LogOP is shown in Fig. 5. The LogOP has reached a optimum of 0.8932, which is the lowest of the three aggregation methods analyzed. For the LogOP aggregation the optimal weights are mostly lower than the average. Since the weights are in the interval $[0, 1]$ and they are used in an exponent, opinions with small weight convert faster against 1.0, thus in the product of the weighted opinions the opinion is ignored.

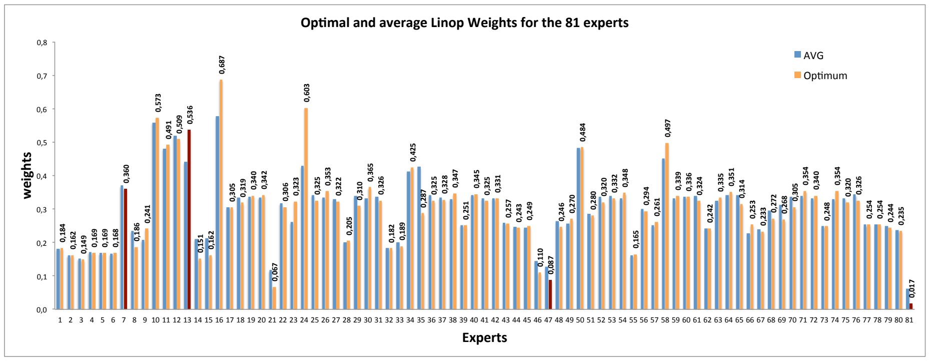

Fig. 6. Weighted arithmetic Mean

The LinOP, shown in Fig. 6, has reached the highest matching assessment of 0.9553 of the three aggregation methods. It can be noticed that the LinOP does tend to the extremes of the weights. This can be seen for instance on agent number 16 and 21, where in both cases the weights are the smallest and the biggest one of the three weight sets. Furthermore the normalization of the weight seams to overcast the result of the learning. Since the sum of the weights are always normalized to 1.0, experts with many siblings do have smaller weights than experts with less siblings. The weights for expert number 1 to 6 are a good example for that, since those are the text similarity experts. It can be noticed that the weight for the different pooling methods are similar. This is in first place because of the bias of the normalization and secondly it is because the experts are weighted due to their performance.

Furthermore all three results of learning the weights seam open to scrutiny since the weights in average underestimate the optimal solution found for big weights and overestimate them when looking at small weights. That behavior is introduced by the the random element in the mutation step of the genetic learning algorithm. Here in small weights, the probability to draw a bigger weight during the leaning is high, thus in average the weight should be higher than the optimal and vice versa with big weights. This can be seen in Fig. 6 with experts 81 and 24.

Overall the weights for the first level of experts can be compared to the result of *Benner* [2] or *Klusch* and *Kapahnke* [13] but it is not a general proposition of weights for semantic service matching since for instance only a few of the services of our test collection have a description of their effect. Thus the weights for the effect matching are low. In other domains the textual description of the service for instance might be neglected, which results in an reduction of the text similarity experts weights.

The logical conclusion is to use an LinOP as an pooling method since it yielded the best results. But we are further intrigued by the different properties of the pooling methods and will analyze further aggregation methods in different domains.

5 Conclusion

We have created an multi-agent expert system realizing a service matcher, where every expert estimates an similarity measure of a service request to a service advertisement. This matching process is seen as an action selection step in an agent planning algorithm. Each expert postulates an opinion concerning the match regarding a part of the service description and advertisement. The different expert opinions are aggregated using pooling methods. The main contribution of this article is the experimental evaluation of different pooling methods namely the Linear and Logarithmic Opinion Pool and the Weighted Harmonic Mean. For this evaluation the weights of the pooling methods are learned using an genetic algorithm and the result are evaluated using a standard service collection. Regarding the overall performance the weighted arithmetic mean seams favorable for this service collection since it produced the best overall matching result of an NDCG value of 0.9553.

With this approach an agent might adapt its action selection process to the domain of use resulting in a performance increase during the planning. The SeMa^2 has been used and extended in an software development methodology in [7] by using service planning to create service compositions. Selecting one service to satisfy a query assumes that this given query has been foreseen and a corresponding service has been implemented. Without loss of generality we assume that this is not always the case, making it necessary to compose multiple services to fulfill a query. Thus for future work the challenge of utilizing such an service matcher in an planning algorithm remains. In future work we will integrate this selection process into the JIAC V agent framework (see: http://www.jiac.de)

References

1. Benediktsson, J.A., Swain, P.H.: Consensus theoretic classification methods. Systems, Man and Cybernetics, IEEE Transactions on 22(4), 688–704 (1992)
2. Bener, A.B., Ozadali, V., Ilhan, E.S.: Semantic matchmaker with precondition and effect matching using SWRL 36(5), 9371–9377 (2009)
3. Beyerer, J.: Verfahren zur quantitativen statistischen Bewertung von Zusatzwissen in der Messtechnik, VDI Fortschritt-Bericht, vol. 8. VDI/Verl. (1999)
4. Della Valle, E., Cerizza, D., Celino, I.: The mediators centric approach to automatic web service discovery of glue. MEDIATE2005 168, 35–50 (2005)
5. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer-Verlag Berlin Heidelberg (2007)
6. Fähndrich, J.: Analyse von Verfahren zur Kombination von Expertenwissen in Form von Wahrscheinlichkeitsverteilungen im Hinblick auf die verteilte lokale Bayes'sche Fusion. Ph.D. thesis, Karlsruhe Institut of Technology (May 2010)
7. Fähndrich, J., Küster, T., Masuch, N.: Semantic Service Management and Orchestration for Adaptive and Evolving Processes. International Journal on Advances in Internet Technology 9(4), 75–88 (2016)
8. Fähndrich, J., Masuch, N., Yildirim, H., Albayrak, S.: Towards Automated Service Matchmaking and Planning for Multi-Agent Systems with OWL-S – Approach and Challenges. In: Service-Oriented Computing – ICSOC 2013 Workshops, pp. 240–247. Springer International Publishing, Cham (Jan 2014)

9. Genest, C.: Pooling operators with the marginalization property. The Canadian Journal of Statistics/La Revue Canadienne de Statistique 12(2), 153–163 (1984)
10. Genest, C., Weerahandi, S., Zidek, J.V.: Aggregating opinions through logarithmic pooling 17(1), 61–70 (Jun 1984)
11. Klusch, M., Fries, B., Sycara, K.: OWLS-MX: A hybrid Semantic Web service matchmaker for OWL-S services. Web Semantics: Science, Services and Agents on the World Wide Web 7(2), 121–133 (Apr 2009)
12. Klusch, M., Gerber, A., Schmidt, M.: Semantic web service composition planning with owls-xplan pp. 55–62 (2005)
13. Klusch, M., Kapahnke, P.: The iSeM matchmaker: A flexible approach for adaptive hybrid semantic service selection 15, 1–14 (Sep 2012)
14. Klusch, M., Kapahnke, P., Zinnikus, I.: SAWSDL-MX2: A Machine-Learning Approach for Integrating Semantic Web Service Matchmaking Variants. In: 2009 IEEE International Conference on Web Services (ICWS). pp. 335–342. IEEE Computer Society, IEEE (2009)
15. Klusch, M., uster, U.K., Leger, A., Martin, D., Paolucci, M.: 5^{th} International Semantic Service Selection Contest - Performance Evaluation of Semantic Service Matchmakers (Nov 2012)
16. Lamparter, S., Ankolekar, A.: Automated selection of configurable web services. 8. Int. Tagung Wirtschaftsinformatik (2007)
17. Masuch, N., Hirsch, B., Burkhardt, M., Heßler, A., Albayrak, S.: SeMa^2: A Hybrid Semantic Service Matching Approach. In: Semantic Web Services, pp. 35–47. Springer Berlin Heidelberg (2012)
18. Morris, P.A.: Combining expert judgments: A Bayesian approach. Management Science 23(7), 679–693 (1977)
19. Sbodio, M.L.: SPARQLent: A SPARQL Based Intelligent Agent Performing Service Matchmaking. pp. 83–105. Springer Berlin Heidelberg, Berlin, Heidelberg (Apr 2012), `http://link.springer.com/10.1007/978-3-642-28735-0_6`
20. Shafer, G.: A mathematical theory of evidence, vol. 1. Princeton university press (1976)
21. Stone, M.: The opinion pool. The Annals of Mathematical Statistics 32(4), 1339–1342 (1961)
22. Wu, J., Wu, Z.: Similarity-based web service matchmaking. Services Computing (2005), `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1531265`
23. Wu, Z., Palmer, M.: Verbs Semantics and Lexical Selection. In: Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics. pp. 133–138. Association for Computational Linguistics, Stroudsburg, PA, USA (1994)