Effective Service Discovery based on Pertinence Probabilities Learning

Mohammed Merzoug, Abdelhak Etchiali, Fethallah Hadjila, Amina Bekkouche
Computer Science Department
Abou Bekr Belkaid University of Tlemcen
B.P. 119 Faculty of Sciences, Tlemcen, Algeria

***Abstract*—Web service discovery is one of the most motivating issues of service-oriented computing field. Several approaches have been proposed to tackle this problem. In general, they leverage similarity measures or logic-based reasoning to perform this task, but they still present some limitations in terms of effectiveness. In this paper, we propose a probabilistic-based approach to merge a set of matching algorithms and boost the global performance. The key idea consists of learning a set of relevance probabilities; thereafter, we use them to produce a combined ranking. The conducted experiments on the real world dataset "OWL-S TC 2" demonstrate the effectiveness of our model in terms of mean averaged precision (MAP); more specifically, our solution, termed "probabilistic fusion", outperforms all the state of the art matchmakers as well as the most prominent similarity measures.**

***Keywords*—Service-oriented computing; web service discovery; rank aggregation; probabilistic fusion**

I. Introduction

The web service technology is actually involved in many applications, such as business processes management and recommendation systems [1]

Thanks to its modularity, composability and loose coupling, this technology is largely utilized in data integration and applications' composition. To ensure these objectives, one has to discover and rank the services that best meet her/ his needs. According to [2], the service discovery can defined as follows:

> Given a web service repository, and a query requesting a service (hereafter service query), finding automatically a service from the repository that matches these requirements is the web service discovery problem. Only those services that: 1) produce at least the requested output parameters that satisfy the post-conditions, 2) use only part of the provided input parameters that satisfy the pre-conditions, and 3) produce the same side effects can be valid solutions to the query.

Several approaches have been proposed in the literature for tackling the web service discovery problem [3]. Based on the works of [4], [5], we distinguish three types of discovery approaches: logic-based reasoning methods, non logic-based techniques (i.e. similarity measures, graph matching, datamining, etc.) and hybrid techniques which merge the logic and the non logic solutions. Despite the progress made in this field, much remains to be done to achieve an acceptable rate of performance. For instance, the logic-based approaches are often characterized by a poor recall rate (Since the underlying semantic of service interfaces can be implicit and not captured by the ontologies) [4]. On the other hand, the similarity measures do not have the same performance; in addition, the choice of the most relevant similarity is not obvious and generally it depends on the actual user's request. Furthermore, a lot of similarity measures may have hyper-parameters (e.g. the fuzzy similarity proposed in [6]) that need to be adjusted for the search; therefore, arbitrary initialization of these parameters is inappropriate and may entail misleading results. Consequently, we must utilize both types of matching algorithms to enhance the discovery performance. In this line of thought, the creation of a hybrid matching algorithm must address the following concerns:

1) How to solve the ordering conflicts entailed by the individual matching algorithms (for instance an algorithm may conclude that service S_1 is better than service S_2, while another may decide that S_2 is better than S_1)?
2) How to infer the most suitable matching algorithm for each user's request, and exploit this knowledge in the fused scheme?
3) How to boost both recall and precision, while preserving a tolerable execution time?

In this paper, we handle the aforementioned difficulties, by adopting machine learning and the theory of probability as a clue for combining the individual matching algorithms.

More specifically, given the m rankings provided by the m matching algorithms (or similarity measures), our machine learning algorithm derives a global ranking by calculating a fusion score for each service S_i; this score is weighted sum of the scores (denoted as $score_{ij}$ where j is the identifier of a matching algorithm) provided by the matching algorithms. Each $score_{ij}$ represents the probability that S_i is relevant to the current request; the more the value of $score_{ij}$ is high, the better the fusion score of S_i. With this fusion scheme, we can answer the abovementioned concerns. In particular, the ordering conflicts are resolved using the weighted sum (which can be considered as weighted vote). Additionally, the most suitable matching algorithms are those that have a higher weight and a higher value of $score_{ij}$ (see equation 22). These heuristic will ensure a good performance in terms of recall and precision. Moreover, if we assume that the m matching algorithms are independent and have a precision equal to p (where $p >= 0.5$), then according to the theorem of jury [7], a majority voting method (or a weighted voting method) will achieve a precision higher than p. In summary, our proposed solution is can be described as follows:

First, we divide each individual ranking into a set of segments. Second, for each segment, we compute its probability of relevance (i.e the probability of having a relevant segment member with respect to the current request). Third, we aggregate the aforementioned probabilities through a linear formula. To choose the ideal number of segments (ns) used in the second step, we perform a cross-validity that evaluate the mean averaged precision of the proposed model.

The remainder of the paper is organized as follows. In Section II, we review the state of the art. We formally define the problem in Section III. Section IV presents the probabilistic fusion algorithm. The results of the experimental study and threads to validity are presented in Section V. Finally, Section VI concludes the paper.

II. State of the Art

The web service discovery has received much attention in the recent years. In general, we discern three types of web service matchmaking approaches: logic matchmaking, non logic matchmaking and hybrid matchmaking [5], [3], [8].

A. Logic-based Matchmaking

The first category of matchning leverages pure logic reasoning, more precisely, the matchmaking utilizes the consistency tests or subsumption mechanisms to decide whether a relationship exists between the user request and the advertised service [9].

The work by [10] presents an automatic location of services (ALS) that allows for discerning five magnitudes of matching degrees (Match, ParMatch, PossMatch, NoMatch and PossParMatch).

In [11], the authors enhance the framework proposed in [10]; in particular, they add additional magnitudes of matching degree such as:

- *RelationMatch*: The advertised service does not meet the required outputs, but it offers outputs having a relation with them.
- *ExcessMatch*: The advertised service meets all the required outputs, but it offers supplementary outputs that are not needed by the user.

A logical matching framework is presented in [9]; this latter architecture takes into account almost all functional properties, including inputs, outputs, preconditions, and effects (IOPE).

-The major weakness of logic-based approaches are the high rate of false positives and false negatives [4].

In addition, the theoretical complexity of subsumption test is Pspace-complete or exp-time complete for certain portions of description logics [12].

B. Non-logic-based Matchmaking

Based on the fact that the aforementioned pitfalls discourage the research in this type of matchmaking, some scholars have developed a new type of solutions. These techniques [13] mainly leverage graph matching, data-mining, combinatorial optimization, and probabilistic matching.

The framework proposed by [8] matches the user's request against the OWL-S using the parameters of service name, service input, and service output. These attributes are first filtered using the part of speech (POS) procedure to eliminate the Stop Words, Special Characters, Numbers, and Uncategorized nouns. Then, the resulting terms are disambiguated using the Wordnet directory. At the end, these terms are matched using a Wordnet based similarity measure.

A new redescription of services is presented in [14]. The main idea consists of using dischlit probability distributions[15] and clustering [16] to provide a latent factor-based specification of services.

The iMatcher1 framework presented in [17] leverages the service profile to perform a syntactic matching of services ; more specifically, it uses four distance functions to match the request and the services (Term Frequency-Inverse Document Frequency [18], the Levenshtein similarity distance [19], the Cosine vector measurement [20], and the divergence measurement of Jensen-Shannon [21]).

In [22], the authors utilize fuzzy sets and rule based systems to tackle the web service discovery and selection problem. More specifically, the proposed work matches both capability attributes (functional aspect) and context attributes (non-functional aspect).

The work by [23] presents a collective dominance function to handle the QoS preferences of a set of users. This function is more flexible and enables the controle of the size of the service skylines.

In [24], the authors tackle the discovery of services. While taking into account the dynamic QoS properties. In particular, they leverage statistical time series to model the QoS fluctuations.

The work in [25] defines a composition framework by means of integration with fine-grained I/O service discovery that enables the generation of a graph-based composition which contains the set of services that are semantically relevant for an input-output request. The proposed framework also includes an optimal composition search algorithm to extract the best composition.

The work of [26] compare the semantic discovery approaches according to several criteria, such as interface type (e.g. OWL-S, WSMO), the scalability, the request expansion, the adopted similarity measure and the use of natural language processing.

The work by [27] proposes a two-stage discovery approach: an offline phase and an online phase. The input of the offline phase is a set of categorized services (most of the existing registries ensure this categorization (e.g. Programmable Web)). Each service is represented as a set of service goals. A service goal is a triple constituted of a verb, a core noun and optional parameters (such as adjectives and non-core nouns). The ensemble of service goals extracted from all services of each category are clustered into groups using K-means algorithm and Wordnet-based similarity measure.

In the online phase, the nearest category (with respect to the request) is retained and thereafter the user's request is expanded using the service goal clusters of the previous

category. At the end, the services of the target category are matched against the expanded query.

C. Hybrid Matchmaking

The third class aggregates the former categories in order to enhance the search quality. There are several ways for merging the aforementioned types: either by using machine learning or heuristics to tune the weights of the matching algorithms, or by using social choice theory to fuse the input rankings, or by leveraging probabilistic / fuzzy relationships to ensure the same purpose.

The most simple heuristic for merging a set of individual matching algorithms is to associate a fixed rank t (or priority) to each matching function.

The OWLS-MX framework [28] matches the inputs/outputs attributes of service profiles. This system proposes seven levels of matching degree (Plug-in, Subsumes and Subsumed-By) and hybrid matching (Logic-based Fail and Nearest-neighbor).

The work by [29] introduces a matchmaker for SAWSDL-based services. The approach leverages both subsumption test and information retrieval models for pairing the request and the advertised services.

The ISEM framework [5] is a hybrid matching approach that combines both the OWLS-MX3 filters and SVM-based learning for discovering services.

Merzoug and al. [3], fuse five matching algorithms (i.e. similarity measures) using a fuzzy dominance relationship [30].

In [31] the authors develop three probabilistic functions for searching and ranking web services. Each function involves multiple matching algorithms (logic, textual similarities, etc.).

In the same work [31], the authors show a comparative evaluation which involves several voting models, such as CombSUM, CombMNZ [32], Borda-fuse model [33], and outranking model [34]. According to the experiments, the CombMNZ system is better than the other voting models, but it is less effective than some individual matching algorithms (such as information loss).

In [35], the authors adapt also the Condorcet fuse model [36] to the service discovery problem. More specifically, they compare the partial scores provided by the individual matching functions through a fuzzified version of the dominance relationship [6]. The preliminary results show that the proposed approach largely outperforms the individual algorithms. However, the results can be largely boosted if a smart parameter tuning is performed.

In [37], the authors introduce a new context-based solution based on QoS (Quality of Service) exploiting both functional and non-functional user's requirements and providing the user ability to control and proceed with the discovery of web services, i.e. the main aim of this work is to locate the appropriate web service correspondence with the context of the user.

In [38], the authors propose a multi-criteria decision method (MCDM) for searching web services based on contextual attributes (e.g. location, language, and size of screen).

TABLE I. EXAMPLES OF PARTIAL MATCHING SCORES OF SERVICES

Services	Parameter	f_1	Services	Parameter	f_2
A	$score_{in}$	0.78	*B*	$score_{in}$	0.86
	$score_{out}$	0.84		$score_{out}$	0.80
	$mean$	0.81		$mean$	0.83
B	$score_{in}$	0.76	*A*	$score_{in}$	0.86
	$score_{out}$	0.80		$score_{out}$	0.78
	$mean$	0.78		$mean$	0.82
C	$score_{in}$	0.78	*C*	$score_{in}$	0.74
	$score_{out}$	0.60		$score_{out}$	0.62
	$mean$	0.69		$mean$	0.68

Since the standard similarity measures (such as Cosine and Extended Jaccard) are not suitable for handling contextual attributes, the authors propose a set of rules and a voting method to compare and rank services.

III. PROBLEM STATEMENT

A. Introduction

In the following, we present a motivating scenario that highlights the major difficulties encountered in web service discovery. We assume that a given user is interested by a service which accepts a set of input concepts $Pin_1, Pin_2...$ and provides a set of output concepts $Pout_1, Pout_2...$, (for the sake of simplicity we disregard for the moment, the other attributes such as preconditions or effects).

To achieve this purpose, the customer may utilize multiple matchmaking algorithms or similarity functions denoted by $f_1 \cdots f_n$. Each function is applied on the request/service parameters (in our case the inputs/ outputs). Let RQ be the request parameter set, i.e. $RQ = RQ_{in} \cup RQ_{out}$, where $RQ_{in} = \{Pin_1, Pin_2, ...\}$, $RQ_{out} = \{Pout_1, Pout_2, ...\}$. Similarly we define the parameter set of the advertised service S as follows: $AS = AS_{in} \cup AS_{out}$.

Each matchmaking function f_j matches the request parameters against the parameters of the advertised services by applying the following equations.

$$score_{in} = f_j(RQ_{in}, AS_{in}) \quad (1)$$

$$score_{out} = f_j(RQ_{out}, AS_{out}) \quad (2)$$

Equations 1, 2 compute the similarity degree between the inputs (resp outputs) of the request and the inputs (resp outputs) of the advertised service.

Table I shows two ranked lists produced by two matching functions f_1 and f_2. Each cell labelled with $score_{in}$ or $score_{out}$ indicates a partial matching score computed through Equations 1 and 2. These matching scores belong to [0,1]. The aforementioned (individual) lists are ranked according to the mean score.

For the sake of simplicity, we suppose that all services have a single input Pin and a single output Pout, the same assumption is considered for the request. By analysing the previous table, we notice the following findings:

First, the two rankings disagree about the ordering of the services A and B. Second, the resolution of the conflict by

computing the mean score over all partial matching scores (see the third line of each service) is not always a relevant heuristic. This solution may be erroneous for some user's requests.

Thus, the creation of an optimal ranking (which provides the highest precision and recall) is not obvious, since we must deal with the specificities of each request as well as the service position within each (individual) list.

As discussed above, each matching function is only effective on a subset of requests, and it may give a poor performance on the remaining requests. Consequently, it will be advantageous to combine a set of matching functions. By doing so, we leverage the advantages of the adopted matching techniques, and we boost the global performances.

To combine the individual matching algorithms, we have to aggregate the partial scores/ ranks of the services. Several aggregating schemes are proposed in the literature [28], [3]. These approaches may leverage voting based models, probability theory, fuzzy set theory, and machine learning.

To determine the most effective mechanism, we have to conduct an exhaustive comparative study and derive the optimal configuration of parameters.

B. Specification of the Discovery Problem

To facilitate the presentation of the problem, we assume the following notations:

let PRL_{ij} be a (partially) ranked list of the i^{th} request under the j^th matching function.

Formally:

$$PRL_{ij} =< (S_1, V_{1ij}), (S_2, V_{2ij}), ...(S_{|dataset|}, V_{|dataset|ij}) >$$

where, $dataset$ is the collection of services (i.e $S_1, .., S_{|dataset|}$) and $V_{kij} \in R^d$, each V_{kij} represents a partial matching score computed through Equation 1 or Equation 2. It measures the similarity between the parameters of the i^{th} request and the parameters of the k^{th} service using the j^{th} matching function. In this case, d is set to 2, since we have two descriptors for inputs and for outputs.

In the following, we specify the discovery problem as follows. Given:

- A set of matching functions $\{f_1, ...f_m\}$.
- A set of user's requests $Q = \{RQ_1, RQ_2, ...\}$; each request is represented by the union of the inputs concepts and the outputs concepts, i.e. $RQ_i = \{Pin_1, Pin_2, ...\} \cup \{Pout_1, Pout_2...\}$.
- A set of (partially) ranked lists, for each request $\{PRL_{11}, ..., PRL_{m1}, .., PRL_{1|Q|} \ldots, PRL_{m|Q|}\}$

We aim to produce a combined ranking (denoted $Combined\_Ranking_i$) for each request RQ_i, such that:

MAP($Combined\_Ranking_1, .., Combined\_Ranking_{|Q|}$) is maximized.

Where:

$Combined\_Ranking_i$: represents the fused list of the i^th request (RQ_i).

MAP: represents the mean average precision criterion. It is defined as follows:

$$MAP(Combined\_Ranking_1, .., Combined\_Ranking_{|Q|}) = \frac{1}{\sum_{i=1}^{|Q|} AveragePrec(Combined\_Ranking_i)} \quad (3)$$

and

$$AveragePrec(Combined\_Ranking_i) = \sum_{k=1}^{|dataset|} precision(Combined\_Ranking_i, k) * rel(k) \quad (4)$$

where $precision(Combined\_Ranking_i, k)$ is the precision at the k^{th} position over the i^{th} combined ranking.

and

$$rel(k) = \begin{cases} 1 & \text{if the service } S_k \text{ is relevant to the } i^{th} \text{ request} \\ 0 & \text{Otherwise} \end{cases} \quad (5)$$

IV. Web Service Discovery and Ranking

In what follows, we present our main contributions to solve this service discovery problem; in particular, we demonstrate the individual matching algorithms (Sections IV-A) as well as the probabilistic fusion scheme (Section IV-B).

A. Individual Matching Functions

In this work, we use the most promising matching functions of the information retrieval field. More specifically, we use five matching functions that are defined below. To match a request R with the service S, we introduce the following notation:

Let RQ be the parameters set of R. let V_{ir} (resp V_{or}) be the vector containing the occurrence numbers of the indexed inputs (resp outputs) of the request R. V_{ir} is derived from RQ_{in}; similarly, V_{or} is derived from RQ_{out}.

In addition, let V_{is} (resp V_{os}) be the vector containing the occurrence numbers of the indexed inputs (resp outputs) of the service S. V_{is} is derived from AS_{in} , similarly V_{os} is derived from AS_{out}. Based on the aforementioned vectors, we define the probability distributions P_{ir} (resp P_{or}) and P_{is} (resp P_{os}) as follows:

$$P_{ir}(k) = \frac{V_{ir}(k)}{\sum_{k=1}^{|V_{ir}|} V_{ir}(k)} \quad (6)$$

$$P_{is}(k) = \frac{V_{is}(k)}{\sum_{k=1}^{|V_{is}|} V_{is}(k)} \quad (7)$$

$$P_{or}(k) = \frac{V_{or}(k)}{\sum_{k=1}^{|V_{or}|} V_{or}(k)} \quad (8)$$

$$P_{os}(k) = \frac{V_{os}(k)}{\sum_{k=1}^{|V_{os}|} V_{os}(k)} \quad (9)$$

The first similarity measure is defined as follows:

$$sim1(R,S) = \frac{1}{2}(cos(V_{ir}, V_{is}) + cos(V_{or}, V_{os})) \quad (10)$$

where cos measures the proportion between the dot product of the compared vectors (or objects) and the product of their length. It is defined as follows :

$$cos(V_{ir}, V_{is}) = \frac{< V_{ir}, V_{is} >}{(||V_{ir}||.||V_{is}||)} \quad (11)$$

and $< X, Y >$ is the dot product operator, $||X||$ is the euclidean norm of X.

Similarly, for V_{or} and V_{os}:

$$sim2(R,S) = \frac{1}{2}(EJ(V_{ir}, V_{is}) + EJ(V_{or}, V_{os})) \quad (12)$$

where EJ (Extended Jaccard) computes the proportion between the size of shared elements and the cardinal of the union. It is defined as follows:

$$EJ(V_{ir}, V_{is}) = \frac{< V_{ir}, V_{is} >}{(||V_{ir}||^2 + ||V_{is}||^2 - < V_{ir}, V_{is} >)} \quad (13)$$

Similarly, for V_{or} and V_{os}:

$$sim3(R,S) = \frac{1}{2}(IL(V_{ir}, V_{is}) + IL(V_{or}, V_{os})) \quad (14)$$

where IL (Information Loss) is based on the percentage of elements that are not shared among the compared objects. The more the percentage is low, the better the similarity degree. It is defined for binary vectors as follows:

$$IL(V_{ir}, V_{is}) = 1 - [\frac{(\sum_{k=1}^{|V_{ir}|} MAX(V_{ir}(k), V_{is}(k)) - < V_{ir}, V_{is} >)}{(\sum_{k=1}^{|V_{ir}|} V_{ir}(k) + \sum_{k=1}^{|V_{is}|} V_{is}(k))}] \quad (15)$$

Similarly, for V_{or} and V_{os}:

$$sim4(R,S) = \frac{1}{2}(JS(P_{ir}, P_{is}) + JS(P_{or}, P_{os})) \quad (16)$$

where JS (Jensen–Shannon based similarity) is based on the estimation of the difference between two probability distributions that represent the compared vectors. The more the difference is low, the better the similarity degree. It is defined as follows:

$$JS(P_{ir}, P_{is}) = (\frac{1}{2}log2) * \sum_{k=1}^{|P_{ir}|} [h(P_{ir}(k)) + h(P_{is}(k)) - h(P_{ir}(k) + P_{is}(k))] \quad (17)$$

where $h(x) = -xlog2(x)$.

Similarly, for P_{or} and P_{os}:

$$sim5(R,S) = \frac{1}{2}(LOG(AS_{in}, RQ_{in}) + LOG(RQ_{out}, AS_{out})) \quad (18)$$

Where LOG (logic matching) is defined as follows :

$$LOG(RQ_{out}, AS_{out}) = MIN_{P_l \in RQ_{out}}(LogMatch1(P_l, As_{out})) \quad (19)$$

In addition:

$$LogMatch1(P_1, AS_{out}) = MAX_{P_k \in AS_{out}}(LogMatch2(P_l, P_k)) \quad (20)$$

In general, the logical comparison of two parameters P_u, P_t is established as follows:

$$LogMatch2(P_u, P_t) = \begin{cases} 1(Exact) & \text{if } P_u \equiv P_t \\ 0.95(plugin) & \text{if } P_u \text{ is parent of } P_t \\ 0.85(Subsume) & \text{if } P_u \sqsubset P_t \\ 0.75(Subsumedby) & \text{if } P_t \text{ is a parent of } P_u \\ 0(Fail) & \text{Otherwise} \end{cases} \quad (21)$$

This is done similarly for AS_{in} and RQ_{in}.

In the following, we present our probabilistic fusion scheme, which is constituted of 3 algorithms. The first one, hereafter referred to as **RPC** (Relevance Probability Computation), computes the knowledge that allows the fusion of the input lists. The second one is termed **PF** (probabilistic fusion), it produces the TopK elements of the combined (fused) ranking. The third one is termed **CVBT** (cross-validation-based tuning). CVBT leverages the cross-validation to select the optimal number of segments.

B. Proposed Algorithms

To build the combined ranking, we adapt the probabilistic approach proposed in [39], to the context of web services. In a nutshell, the basic idea consists of learning a set of probabilities that are involved in the computation of the fused score of each service. The more the fused score is high, the better the rank is. The algorithm performing this task is referred to as RPC. Each learned probability (denoted by $MRelP_{ri}(S_l)$) represents the likelihood that a service S_l returned in segment r is relevant, given that it has been returned by the matching function i.

Algorithm 1 represents the pseudo code of RPC.

RPC algorithm is explained as follows:

Algorithm 1: Algorithm RPC

Input: $dataset : a\ set\ of\ services$,
$SRQ : a\ subset\ of\ requests$,
$m : Integer\ (the\ number\ of\ matching\ functions)$,
$ns :\ Integer\ (the\ number\ of\ segments)$,
$Rel :$
$a\ binary\ matrix\ of\ dimension\ |requests|\ .\ |dataset|$
Output: $MRelP :\ a\ matrix\ of\ dimension\ ns.m$

1 **for** $i = 1\ to\ m$ **do**
2 $\quad ranking_i = EmptyList$
3 $\quad$ **foreach** $Q_j \in SRQ$ **do**
4 $\quad\quad$ **foreach** $S_l \in dataset$ **do**
5 $\quad\quad\quad score = sim_i(Q_j, S_l)$
$\quad\quad\quad\ insertInto(score, ranking_i)$
6 $\quad\quad decreasing\_sort(ranking_i)$
7 $\quad\quad relv\_services = extract(Rel, j)$
8 $\quad\quad$ **for** $r = 1\ to\ ns$ **do**
9 $\quad\quad\quad segment\_members = extract\_seg(ranking_i, r)$
10 $\quad\quad\quad relv[i][j][r] =$
$\quad\quad\quad\quad |relv\_services\_segment\_members|/|segment\_members|$
11 $\quad$ **for** $r = 1\ to\ ns$ **do**
12 $\quad\quad$ **for** $b = 1\ to\ |SRQ|$ **do**
13 $\quad\quad\quad MRelP[i][r] = MRelP[i][r] + relv[i][b][r]$
14 $\quad\quad MrelP[i][r] = MRelP[i][r]/|SRQ|$
15 $return\ (MRelP)$

- (Lines 1 up to 7), for each matching function i and request Q_j, we compute the corresponding ranking termed $ranking_i$.
- (Lines 8-9), we sort the aforementioned ranking and we get the relevant services of the request Q_j.
- (Lines 10-13), for each segment r, we extract its members, thereafter we compute its relevance probability by applying the formula of the precision criterion. This rule calculates the likelihood that a segment r derived from the function i is relevant to the request Q_j.
- (Lines 15-19), for each segment r and each matching function i, we compute their averaged relevance probability (also denoted $MRelP_{ri}$). More specifically, we take the mean of the relevance probabilities related to the requests of the learning set (SRQ).
- (line 22) We return the learned probabilities.

The second algorithm referred to as **PF** (probabilistic fusion) allows to compute a fused score for each service S_l. To this end, **PF** leverages the learned relevance probabilities of the five individual rankings. PF is based on two heuristics (H1 and H2); which are summarized as follows:

- The more the rank (or the segment identifier) of a service S_l is higher within the individual rankings, the more the fused score is better (H1). •
- The more the relevance probability $MRelP_{ri}(S_l)$ is higher, the more the fused score is better (H2).
This rule is explained as follows: if we assume that $MRelP_{ri}(S_l)$ is large, then service S_l is more likely to be relevant and, thus it should be ranked higher in the combined (fused) list. The fused score is summed up as follows:

$$Fscore(S_l) = \sum_{i=1}^{m} \frac{MRelP_{ri}(S_l)}{r} \tag{22}$$

where r is the segment identifier of S_l, i the identifier of the matching function, and m is the number of matching functions.

The pseudo code of **PF** is given in Algorithm 2.

Algorithm 2: Algorithm PF

Input: $Q_j : the\ current\ request\ to\ be\ handled$, $m :$
$Integer\ (the\ number\ of\ matching\ functions)$,
$k :\ Integer\ (size\ of\ the\ returned\ list)$,
$ns :\ Integer\ (the\ number\ of\ segments)$,
$MRelP : a\ matrix\ of\ dimension\ m.ns$
Output: $Top\_k(CombinedList) :\ an\ ordered\ list$

1 **for** $l = 0\ to\ |dataset| - 1$ **do**
2 $\quad Fscore[l] = 0$
3 $CombinedList = EmptyList$
4 **for** $i = 1\ to\ m$ **do**
5 $\quad ranking_i = EmptyList$
6 $\quad$ **for** $each S_l\ in\ dataset$ **do**
7 $\quad\quad score = sim_i(Q_j, S_l)$
8 $\quad\quad insertInto(score, S_l, ranking_i)$
9 $\quad decreasing\_sort(ranking_i)$
10 $\quad$ **for** $each\ S_l\ in\ dataset$ **do**
11 $\quad Sid = Get\_Seg\_ID(S_l, ranking_i)$
12 $\quad Fscore[l] = Fscore[l] + MRelP[i][sid]/sid$
13 **for** $each S_l\ in\ dataset$ **do**
14 $\quad insertInto(Fscore[l], S_l, CombineList)$
15 $decreasing\_sort(CombineList)$
16 $return\ Top\_k(CombinedList)$

PF algorithm is explained as follows:

- (Lines 1-4), we initialize the fused scores with 0, the fused (combined) ranking is also initialized with an empty list.
- (Lines 5-10) for each matching function i and the current request Q_j, we compute the corresponding ranking termed $ranking_i$.
- (Line 11), we sort the aforementioned ranking.
- (Lines 12-14), for each service S_l, we get the identifier of the segment in which he lies (Sid).
- (Line 15), we update the fused score by applying the formula 22 (heuristics $H1\&H2$).
- (Lines 17-20), we create and sort the combined list, according to the decreasing order of the fused score.
- (Line 21): we return the Top_K elements of the combined list.

In what follows, we present the third algorithm referred to as **CVBT** (cross-validation based tuning). This algorithm

aims to select the optimal number of segments (denoted by ns) that ensures the best mean averaged precision of the combined rankings. We notice that $ns \in \{2, 3, ..., round(|dataset|/2)\}$.

To fulfill this goal, we use the cross-validation principle. This means that we firstly initialize ns to a given value, then we divide the requests collection on a set of parts (np parts). Thereafter, we perform the cross-validation section as follows:

- We compute the relevance probabilities (i.e the RPC function) by choosing $(np - 1)$ parts as the set of learning requests.
- We perform the probabilistic fusion (PF) over the entire set of requests.
- We calculate the mean averaged precision (MAP) that corresponds to the actual learning requests.
- We change the set of learning requests, by considering another union of $(np - 1)$ parts, and we redo the previous steps
- We take the average of the calculated MAP and we consider it as the final MAP associated to the actual ns (we denote this result as MAP'). We iterate the previous process (the five steps) for all possible values of ns and then we choose the value that ensures the best MAP'.

In the following, we describe the pseudo code of **CVBT** (see Algorithm 3)

- (Line 1), we divide the entire collection requests on a set of parts, for instance, if $np = 5$, then we have five subsets of requests.
- (Line 2), we initialize the optimal number of segments, as well as the optimal MAP.
- (Lines 3-6), for each candidate ns, we initialize its corresponding (averaged) MAP' with 0.
- (Lines 7-9), for each iteration of the cross-validation, we initialize the learning requests (SRQ). The latter is constituted of $(np - 1)$ parts of the entire collection. Thereafter we use this set (SRQ) to learn the relevance probabilities $(MRelP)$.
- (Lines 10-13), for each request Q_j we produce the fused ranking *CombinedListj*, afterwards, we calculate the corresponding average precision.
- (line 14), we estimate the mean averaged precision of the actual SRQ set.
- (line 16-19), We calculate the mean averaged precision of the cross-validation loop (denoted MAP'). This score is associated to the actual ns.
- (line 20), we return the optimal number of segments $(ns*)$.

In the following, we show a scenario that illustrates the processing performed by the probabilistic fusion (i.e. RPC and PF).

Algorithm 3: Algorithm CVBT

Input: $dataset: \ the \ set \ of \ services$,
$CRQ: \ the \ collection \ of \ requests$, $m:$
$Integer \ (the \ number \ of \ matching \ functions)$,
$np: \ Integer \ (the \ number \ of \ parts)$,
$the \ defaut \ value \ is \ 5$,
$Rel:$
$a \ binary \ matrix \ of \ dimension \|requests\|.|dataset|$,
$Q_j: \ the \ current \ request \ to \ be \ handled$, $m:$
$Integer \ (the \ number \ of \ matching \ functions)$

Output: $ns*:$
$Integer \ (the \ optimal \ number \ of \ segments)$

1 $Parts = divide(CRQ, np)$
2 $MAP* = 0; \ ns* = 2;$
3 **for** $ns = 2 \ to \ round(|dataset|/2)$ **do**
4 $SRQ = CRQ$
5 $MAP'[ns] = 0$
6 **for** $i = 1 \ to \ np$ **do**
7 $SRQ = SRQ - Parts[i]$
8 $MRelP = RPC(dataset, SRQ, m, ns, Rel)$
9 **for** $each \ Q_j \ in \ CRQ$ **do**
10 $CombinedList_j = FP(MRelP, |dataset|, Q_j)$
11 $AP[j] = AveragePrecision(CombinedList_j, Rel)$
12 $MAP[i] = MeanAveragedPrecision(AP)$
13 **for** $i = 1 \ to \ np$ **do**
14 $MAP'[ns] = MAP'[ns] + MAP[i]$
15 $MAP'[ns] = MAP'[ns]/np$
16 $return \ (ns*)$

V. Experimental Study

This section presents our experiments related to the probabilistic fusion as well as the individual matching functions. We also show a comparison with respect to the Borda [33] fusion scheme and other state of the art methods.

A. Evaluation Scheme

To assess the effectiveness and the efficiency of the proposed fusion scheme, we use the test collection *OWLTC V2.2*[1]. The latter contains real-world web service descriptions, extracted mainly from public IBM UDDI registries. As depicted in Table II, the benchmark contains:

1) 1007 service descriptions,
2) 29 sample requests,
3) a manually identified relevance set for each request. This information allows the computation of recall and precision.

Since we set np to 5 (np is the number of parts), then 80% of the request set is utilized for learning the relevance probabilities. In addition, all requests will be used for evaluating *MAP* and some other metrics *($recall@N, Prec@N, R-prec$) defined below:*

[1] http://projects.semwebcentral.org/projects/owls-tc/

TABLE II. OWLSTC V2 TEST COLLECTION

Class	Number of services	Number of requests
Travel	197	6
Education	286	6
Food	34	1
Medical care	73	1
Communication	59	2
Weapon	40	1
Economy	395	12

TABLE III. AVERAGE EXECUTION TIME OF THE PROBABILISTIC FUSION

Average fusion time (PF function)	Average learning time (RPC function)	Sum
13830 ms	13659 ms	27489 ms

TABLE IV. AVERAGE EXECUTION TIME FOR ALL METHODS

Approach	Probabilitic fusion	Borda	Cos	EJ	IL	JS	LOG
Average time	27489 ms	700 ms	28630 ms	26391 ms	21741 ms	27659 ms	14941

- *R-Precision(R-prec or R-P)*: measures the precision after all relevant items have been retrieved [40].
- Precision at N *(Prec@N)*: measures the precision after N items have been retrieved [40].
- Recall at N *(recall@N)*: measures the recall after N items have been retrieved [40].

We also measure the average execution time of the probabilistic fusion, the Borda fuse model and the individual matching functions. Our algorithms have been implemented in Java and the experiments were conducted on a Core I3 1.80 GHz machine with 4GB of RAM, running on Windows7.

In Table III, we show the average execution time of the learning phase (RPC function), the fusion phase (PF function), and the total time. Since the aforementioned algorithms have a polynomial complexity, then they remain scalable for large services datasets.

In Table IV, we compare the performance of the probabilistic fusion with respect to the other approaches. We observe that all running times fluctuate between 21.000 and 29.000 Milli.Sec, except for Borda. The latter exhibits a performance around 700 ms. This is due to the fact that Borda is a simple sum of the service positions. We also notice that, the logic-based approach is more efficient than the other individual methods, because we implemented the subsumption test with a logic "or". This implementation is enabled by a binary encoding scheme inspired from [41]. By coding the ontology with binary words we significantly decrease the subsumption test cost.

Tables V and VI show the behavior of PF for both recall and precision. In general, we observe that the performance rises as the number of segments ns increases (for all values of K). We also notice that the best performance is provided by $ns = 500$.

TABLE V. THE RECALL WITH RESPECT TO ns

ns	TOP 10	TOP 20	TOP 30	TOP 40	TOP 50	TOP 60
100	0.419	0.661	0.777	0.829	0.867	0.908
150	0.425	0.678	0.793	0.85	0.884	0.911
200	0.423	0.679	0.801	0.86	0.898	0.924
251	0.421	0.679	0.815	0.871	0.904	0.931
300	0.426	0.685	0.806	0.868	0.904	0.932
350	0.434	0.691	0.811	0.873	0.911	0.936
400	0.433	0.691	0.811	0.872	0.911	0.936
500	0.432	0.705	0.828	0.892	0.93	0.949

TABLE VI. THE PRECISION WITH RESPECT TO ns

ns	TOP 10	TOP 20	TOP 30	TOP 40	TOP 50	TOP 60
100	0.906	0.746	0.616	0.506	0.432	0.381
150	0.896	0.76	0.624	0.516	0.437	0.382
200	0.893	0.762	0.633	0.524	0.446	0.389
251	0.893	0.768	0.642	0.529	0.449	0.393
300	0.896	0.77	0.636	0.528	0.448	0.391
350	0.917	0.775	0.637	0.531	0.453	0.393
400	0.913	0.775	0.637	0.531	0.453	0.393
500	0.91	0.789	0.652	0.542	0.461	0.398

TABLE VII. MEAN RECALL FOR ALL METHODS

Algorithm	TOP 10	TOP 20	TOP 30	TOP 40	TOP 50	TOP 60
EJ	0.33	0.59	0.73	0.79	0.83	0.85
IL	0.33	0.59	0.7	0.79	0.84	0.86
JS	0.33	0.59	0.72	0.79	0.83	0.86
LOG	0.3	0.46	0.6	0.66	0.72	0.69
COS	0.33	0.59	0.72	0.78	0.82	0.86
PF	**0.43**	**0.70**	**0.82**	**0.89**	**0.93**	**0.94**
BORDA	0.39	0.58	0.73	0.81	0.84	0.9

TABLE VIII. MEAN PRECISION FOR ALL METHODS

Algorithm	TOP 10	TOP 20	TOP 30	TOP 40	TOP 50	TOP 60
EJ	0.81	0.64	0.58	0.51	0.42	0.36
IL	0.81	0.64	0.58	0.5	0.42	0.36
JS	0.8	0.65	0.57	0.49	0.41	0.36
LOG	0.73	0.53	0.48	0.42	0.37	0.31
COS	0.81	0.64	0.57	0.48	0.4	0.36
PF	**0.90**	**0.76**	**0.63**	**0.52**	**0.44**	**0.39**
BORDA	0.83	0.67	0.58	0.5	0.42	0.38

According to Tables VII and VIII, we observe that PF is more effective than the remaining approaches. The PF results are achieved by setting ns to 500. As demonstrated in the experiments, PF largely outperforms the Borda fuse model.

This is due to the fact that Borda is very sensitive to the services with bad individual ranks. Consequently its global performance is unsatisfying. On the other hand, we notice that the four similarity measures $\{Cos, EJ, IL, JS\}$ have almost the same performance. The worst case is achieved by the logic-based approach.

The execution of *CVBT* is shown in Fig. 1. It depicts the

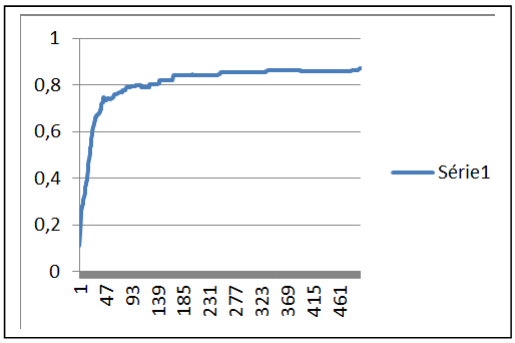

Fig. 1. Mean Average Precision (MAP) vs. ns

TABLE IX. AVERAGE R-PREC FOR ALL METHODS

Algorithm	R-Prec
PF	**0.86**
Borda	0.669
IL	0.709
LOG	0.594
JS	0.705
EJ	0.707
COS	0.692

TABLE X. PF VS S3 C CONTEST APPROACHES

Algorithm	MAP
PF (ns=500	**0.87**
Borda	0.73
OWLS-iMatcher2	0.84
JIAC-OWLSM	0.81
SPARQLent	0.71
OPOSSUM	0.57
ALIVE	0.5
OWLS MX3	0.86

relationship between MAP and ns parameter. In general, we distinguish two behaviours: firstly, when $ns \in \{2, .., 120\}$ we observe a rapid improvement of the estimated MAP. Second, when $ns \in 121, .., 500$, we observe a slow improvement of MAP. The optimal value is reached round 500.

From these results we conclude that the more the segment size is small, the better the performances.

As depicted in Table IX, the $R-Prec$ of PF is higher than the individual ranking algorithms as well as the Borda fuse model. In summary, PF produces a gain of 21% with respect to the highest individual $R - Prec$ (i.e. the information loss R-Prec) and 28% with respect to the Borda $R - Prec$.

Table X shows a comparison between the probabilitic fusion and the different systems that participate in the S3 contest 2009[2]. We notice that this competition is based on the same benchmark (i.e. OWLSTC.2). Table X clearly shows that our approach outperforms all existing matchmakers.

[2]International Contest S3 on Semantic Service Selection 2009, http://www-ags.dfki.uni-sb.de/ klusch/s3/

VI. CONCLUSION

In this paper, we have tackled the problem of retrieving and ranking web services. Our proposed framework takes into account multiple functional descriptors (input and output parameters) as well as several matching functions (logic reasoning and text similarities).

Simply speaking, our fusion algorithm leverages a set of relevance probabilities in order to infer an optimal fused ranking. These probabilities are largely dependent on the number of segments (ns). The setting of this regulating parameter is ensured by the cross-validation process.

The obtained results are very promising, and confirm the effectiveness of the proposed scheme.

In the nearest future, we aim to compare our approach with alternative fusion schemes, such as probabilistic dominance and majority-based voting. These approaches can be further enhanced by tuning their critical parameters with machine learning algorithms.

REFERENCES

[1] S. Sagayaraj and M. Santhoshkumar, "Heterogeneous ensemble learning method for personalized semantic web service recommendation."

[2] S. Kona, A. Bansal, G. Gupta, and T. D. Hite, "Semantics-based web service composition engine," in *Proc. of the 9th IEEE Int. Conf. on E-Commerce Technology (CEC 2007) / 4th IEEE Int. Conf. on Enterprise Computing, E-Commerce and E-Services (EEE 2007)*, 2007, pp. 521–524.

[3] M. Mohammed, C. M. Amine, and H. Fethallah, "Leveraging fuzzy dominance relationship and machine learning for hybrid web service discovery," *International Journal of Web Engineering and Technology*, vol. 11, no. 2, pp. 107–132, 2016.

[4] M. Klusch, B. Fries, and K. Sycara, "OWLS-MX: A hybrid Semantic Web service matchmaker for OWL-S services," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 7, no. 2, pp. 121–133, 2009.

[5] M. Klusch and P. Kapahnke, "The iSeM matchmaker: A flexible approach for adaptive hybrid semantic service selection," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 15, pp. 1–14, 2012.

[6] K. Benouaret, D. Benslimane, A. Hadjali, M. Barhamgi, Z. Maamar, and Q. Z. Sheng, "Web service compositions with fuzzy preferences: A graded dominance relationship-based approach," *ACM Transactions on Internet Technology (TOIT)*, vol. 13, no. 4, p. 12, 2014.

[7] S. Berg, "Condorcet's jury theorem, dependency among jurors," *Social Choice and Welfare*, vol. 10, no. 1, pp. 87–95, 1993.

[8] M. Santhoshkumar and S. Sagayaraj, "Ranking semantic web services by matching triples and query based on similarity measure," *International Journal of Information Technology*, pp. 1–9, 2019.

[9] N. Srinivasan, M. Paolucci, and K. Sycara, "Semantic web service discovery in the OWL-S IDE," in *Proc. of the 39th Annual Hawaii Int. Conf. on System Sciences (HICSS'06)*, vol. 6. IEEE, 2006, pp. 109b–109b.

[10] U. Keller, R. Lara, H. Lausen, A. Polleres, and D. Fensel, "Automatic location of services," *The Semantic Web: Research and Applications*, pp. 1–16, 2005.

[11] U. Küster and B. König-Ries, "Evaluating semantic web service matchmaking effectiveness based on graded relevance," in *Proc. of the 2nd Int. Conf. on Service Matchmaking and Resource Retrieval in the Semantic Web-Volume 416*. CEUR-WS. org, 2008, pp. 32–46.

[12] F. Baader and U. Sattler, "An overview of tableau algorithms for description logics," *Studia Logica*, vol. 69, no. 1, pp. 5–40, 2001.

[13] A. Segev and E. Toch, "Context-based matching and ranking of web services for composition," *IEEE Transactions on Services Computing*, vol. 2, no. 3, pp. 210–222, 2009.

[14] G. Cassar, P. Barnaghi, and K. Moessner, "Probabilistic matchmaking methods for automated service discovery," *IEEE Transactions on Services Computing*, vol. 7, no. 4, pp. 654–666, 2014.

[15] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003. [Online]. Available: http://portal.acm.org/citation.cfm?id=944937

[16] T. Hofmann, "Probabilistic latent semantic analysis," in *Proc. of the 15th Conf. on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1999, pp. 289–296.

[17] M. Schumacher, H. Helin, and H. Schuldt, *CASCOM: intelligent service coordination in the semantic web*. Springer Science & Business Media, 2008.

[18] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[19] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, no. 8, 1966, pp. 707–710.

[20] E. Garcia, "Cosine similarity and term weight tutorial," *Information retrieval intelligence*, 2006.

[21] B. Fuglede and F. Topsoe, "Jensen-shannon divergence and hilbert space embedding," in *Proc. of the Int. Symposium on Information Theory (SIT 2004)*. IEEE, 2004, p. 31.

[22] Z. Chouiref, A. Belkhir, K. Benouaret, and A. Hadjali, "A fuzzy framework for efficient user-centric web service selection," *Applied Soft Computing*, vol. 41, pp. 51–65, 2016.

[23] K. Benouaret, D. Benslimane, and A. Hadjali, "Selecting skyline web services for multiple users preferences," in *Proc. of the 19th IEEE Int. Conf. on Web Services (ICWS'12)*. IEEE, 2012, pp. 635–636.

[24] A. Pahlevan, J.-L. Duprat, A. Thomo, and H. Müller, "Dynamis: effective context-aware web service selection using dynamic attributes," *Future Internet*, vol. 7, no. 2, pp. 110–139, 2015.

[25] P. Rodriguez-Mier, C. Pedrinaci, M. Lama, and M. Mucientes, "An integrated semantic web service discovery and composition framework," *IEEE transactions on services computing*, vol. 9, no. 4, pp. 537–550, 2016.

[26] M. Fariss, N. El Allali, H. Asaidi, and M. Bellouki, "Review of ontology based approaches for web service discovery," in *International Conference on Advanced Information Technology, Services and Systems*. Springer, 2018, pp. 78–87.

[27] N. Zhang, J. Wang, Y. Ma, K. He, Z. Li, and X. F. Liu, "Web service discovery based on goal-oriented query expansion," *Journal of Systems and Software*, vol. 142, pp. 73–91, 2018.

[28] M. Klusch, B. Fries, and K. Sycara, "Automated semantic web service discovery with OWLS-MX," in *Proc. of the 5th Int. Joint Conf. on autonomous agents and multiagent systems*. ACM, 2006, pp. 915–922.

[29] M. Klusch and P. Kapahnke, "Semantic web service selection with SAWSDL-MX," in *Proc. of the 7th Int. Semantic Web Conf.*, 2008, p. 3.

[30] A. Halfaoui, F. Hadjila, and F. Didi, "Qos-aware web service selection based on self-organising migrating algorithm and fuzzy dominance," *International Journal of Computational Science and Engineering*, vol. 17, no. 4, pp. 377–389, 2018.

[31] D. Skoutas, D. Sacharidis, A. Simitsis, and T. Sellis, "Ranking and clustering web services using multicriteria dominance relationships," *IEEE Transactions on Services Computing*, no. 3, pp. 163–177, 2010.

[32] E. A. Fox and J. A. Shaw, "Combination of multiple searches," *NIST special publication SP*, vol. 243, 1994.

[33] J. A. Aslam and M. Montague, "Models for metasearch," in *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '01. New York, NY, USA: Association for Computing Machinery, 2001, p. 276–284. [Online]. Available: https://doi.org/10.1145/383952.384007

[34] M. Farah and D. Vanderpooten, "An outranking approach for rank aggregation in information retrieval," in *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 591–598. [Online]. Available: https://doi.org/10.1145/1277741.1277843

[35] H. Fethallah, B. Amine, and H. Amel, "Hybrid Web Service Discovery Based on Fuzzy Condorcet Aggregation," in *East Europ. Conf. on Advances in Databases and Information Systems*. Springer, 2015, pp. 415–427.

[36] M. Montague and J. A. Aslam, "Condorcet fusion for improved retrieval," in *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, ser. CIKM '02. New York, NY, USA: Association for Computing Machinery, 2002, p. 538–548. [Online]. Available: https://doi.org/10.1145/584792.584881

[37] S. Samir, A. Sarhan, and A. Algergawy, "Context-based web service discovery framework with qos considerations," in *Proc. of the 11th Int. Conf. on Research Challenges in Information Science (RCIS 2017)*. IEEE, 2017, pp. 146–155.

[38] A. D. Eddine and B. F. M'hamed, "Improved multicriteria ranking based web service discovery approach," in *2018 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS)*. IEEE, 2018, pp. 1–6.

[39] D. Lillis, F. Toolan, R. Collier, and J. Dunnion, "Probfuse: a probabilistic approach to data fusion," in *Proc. of the 29th annual int. ACM SIGIR conf. on research and development in information retrieval*. ACM, 2006, pp. 139–146.

[40] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. USA: Cambridge University Press, 2008.

[41] Y. Caseau, M. Habib, L. Nourine, and O. Raynaud, "Encoding of multiple inheritance hierarchies and partial orders," *Computational Intelligence*, vol. 15, no. 1, pp. 50–62, 1999.