Optimizing Semantic Web services ranking using parallelization and rank aggregation techniques

Ioan Toma
University of Innsbruck, Austria.
ioan.toma@sti2.at

Ying Ding
Indiana University, Indiana, USA.
dingying@indiana.edu

Dieter Fensel
University of Innsbruck, Austria.
dieter.fensel@sti2.at

Abstract: The problem of combining many rank orderings of the same set of candidates, also known as the rank aggregation problem, has been intensively investigated in the context of Web (e.g meta-search) databases (e.g combining results from multiple databases), statistics (e.g. correlations), and last but not least sports and elections competitions. In this paper we investigate the use of rank aggregation in the context of Semantic Web services. More precisely we propose an optimization technique for ranking Semantic Web services based on non-functional properties by using parallelization and rank aggregation methods. Instead of using a ranking algorithm over the entire set of non-functional properties our approach splits the set of non-functional properties in multiple subsets, runs the ranking algorithm on each of the subsets and finally aggregates the resulting ranked lists of services into one unifying ranked list. Experimental results reported in this paper show improvements of our initial rank aggregation method both in terms of quality and processing time.

1 Introduction

Two of the most important challenges in any service-oriented infrastructure are how to identify the relevant services given a user request and how to provide an ordered list of services according to user preferences. Known as service ranking, the later challenge has been investigated both by academia and industry ([26], [15], [19]).

Existing approaches for service ranking provide in general one single rank ordering given a set of services and a user request. Based on the same input data (same user request and services) different ranking engines create different rank lists. This is due to specific algorithms employed by each of the ranking engines, different ways of interpreting the request and service descriptions, etc. In the future service-oriented ecosystems, ranking engines become service themselves that produce different rank lists given the same user request. One challenge that becomes important in this context is how to identify the best rank list out of a given set of rank lists produced by different ranking engines. This is a difficult task and in most cases difficult to achieve due to the lack of access to the inner mechanics of the individual ranking engines.

A consensus mechanism that combines the rank lists produces by individual rank engines into a communally agreed rank list is more likely to create the 'optimal' solution. Building such a consensus is equivalent to solving the *rank aggregation* problem. This problem can be defined more formally as follows. Given a set of rank lists R_1,...,R_m, each of them the rank lists of the set S of services $S = \{s_1, s_2, ..., s_n\}$, the problem is to identify the 'optimal' rank list σ such that $\sum_{k=1}^{m} D(\sigma, R_k)$ is minimized, where D is a distance metric showing how much two rank lists are similar respectively different.

The rank aggregation problem has been studied in the context of many fields such as social choice theory ([10], [27]), statistics ([5], [4]), machine learning ([13], [14]), web search engines ([7], [21]),sports and competitions ([11], [24]). To the best of out knowledge the rank aggregation problem has not been investigated in the context of finding consensus among Semantic Web services rank engines.

Semantic Web services [8] are a new paradigm that combines on one hand the *Semantic Web* [9] vision, as the augmentation of the current Web supporting meaningful retrieval of data and interaction in a precise, semantically defined way, and on the other hand *Web services*,as the technology that brings the aspect of dynamic and distributed computation into the current Web. Semantic Web services provide increasing degree of automation with respect to all service related tasks, including discovery, composition, execution and last but not least, *ranking*. Among the properties of services in general, and Semantic Web services in particular, *non-functional properties* are considered to be the most relevant properties for the task of ranking services and identifying the best service given a user request. They describe restrictions over the other properties of the services [2] including functional and behavioural properties.

In this paper we investigate the use of rank aggregation in the context of Semantic Web services. More precisely we propose an optimization technique for ranking Semantic Web services based on non-functional properties by using rank aggregation methods. Instead of using a ranking algorithm that consider the entire set of non-functional properties requested by the user, our approach splits the set of non-functional properties in multiple subsets, runs the ranking algorithm on each of the subsets and finally aggregates the resulting rank lists of services into one unifying rank list.

The rest of the paper is organized as follows. Section 2 introduces some basic concepts used latter on in the paper. A set of rank aggregation methods that have proved to be efficient in other domains (i.e. Web) are described in Section 3. Section 4 discusses our ranking algorithm for Semantic Web services based on non-functional properties. Taking the rank aggregation methods introduced in Section 3, Section 5 proposes optimization techniques for the ranking algorithm introduced in Section 4. Section 6 presents the experimental results obtained for rank aggregation with Semantic Web services. Section 7 discussed related work approaches for the problem of rank aggregation and finally Section 8 concludes the paper and presents our future work.

2 Background

Before we describe our approach for rank aggregation of Semantic Web services we introduce the terminology used in this paper.

Let U be a universe of services, S a subset of U, and $\leq_O$ a partial order relation on S. We define a rank list as an ordering of S on which the partial order relation $\leq_O$ holds over any pair of services. More formally a *rank list* is defined as follows.

[Rank list] Given a set S of services $S = \{s_1, s_2, ..., s_n\}$, a partial order relation $\leq_O$ on this set and a ranking function f, a rank list R can be defined as $R = f(S, \leq_O)$, where R contains all the elements from S and $\exists s$ a sequence $s(1, 2, ..., |R|) \rightarrow R$ such that $\forall i, j \in \{1, 2, ..., |R|\}$, with $i <= j \Rightarrow s(i) \leq_O s(j)$.

In case R contains all the elements from the universe U, R is called a *full rank list*. If only some of the elements of U are ranked, R is call a *partial rank list*.

A *ranking metric* or *ranking distance* is a function that computes the similarity between two rank lists, where the similarity is a positive real number. A ranking metric is defined as follows.

[Ranking Metric] Given two rank lists R_1 and R_2, a ranking metric Δ or distance between the two rank lists is defined as a function $\Delta : \mathcal{R} \times \mathcal{R} \rightarrow \mathbb{R}^+$ that determine the degree of similarity between the two rank lists.

Any *ranking metric* has the following properties: (1) it is *non-negative*, meaning that the distance between two rank lists is always positive, (2) is it *identical*, meaning that if distance between two rank lists is 0 then the two rank lists are the same, (3) it is *symmetric*, meaning that the given two rank lists R_1 and R_2, the distance between R_1 and R_2 is the same as the distance between R_2 and R_1 and finally (4) it preserves the *triangle inequality*, meaning that for any three rank lists R_1, R_2 and R_3 any ranking metric Δ will satisfy the following inequality $\Delta(R_1, R_2) \leqslant \Delta(R_1, R_2) + \Delta(R_2, R_3)$

Some of the most used ranking metrics are *Kendall-Tau* [12] and *Spearman Footrule* [23].

Kendall-Tau (Kendall's τ) determines the degree of similarity between two rank lists by considering the number of inversions of pairs items needed to transform one rank list into the other. It basically corresponds to the number of transpositions *bubble sort* requires to turn one rank list into the other one. The Kendall's τ ranking metric is defined as follows:

$$\tau = 1 - \frac{2\delta}{N(N-1)} \tag{1}$$

where δ is the *symmetric difference distance* between the rank lists created based on the given sets of items and N is the total number of items. The symmetric difference distance is a set operation which associates to the rank lists, the set of elements which belong to rank lists. The Kendall's τ ranking metric value for two lists of length n can be computed in $nlogn$ time [7].

Spearman Footrule (Spearman's ρ) captures how well an arbitrary monotonic function could describe the relationship between two variables, without making any assumptions about the frequency distribution of the variables. The Spearman ranking metric is defined as follows:

$$\rho = 1 - 6\frac{\sum_{i=1}^{N} d_i^2}{N(N^2-1)} \tag{2}$$

where d_i is the difference in statistical rank of corresponding variables, and N is the number of pairs of values. Equation 2 can be rewritten in the following form:

$$\rho(X,Y) = \frac{\sum_{i=1}^{N} x_i y_i}{\sqrt{\sum_{i=1}^{N} x_i^2 \sum_{i=1}^{N} y_i^2}} \tag{3}$$

The Spearman Footrule metric between two lists can be computed in a linear time.

3 Rank aggregation methods

In this section we present the rank aggregation methods that are used to optimize our existing ranking approach for Semantic Web services based on non-functional properties [25].

Borda count method The Borda count [1] is a rank aggregation method that computes the aggregated rank list based on the cumulative positional score obtained by each candidate in each rank list. The candidate with the highest score is the winner, followed by the other candidates in decreasing order of their scores. More formally the Borda count method works as follows. Given the full rank lists $R_1, R_2, ..., R_n$ of a set of services S, then for each candidate $s \in S$, Borda count method assign a score $B_i(s)$ equal to the number of the candidate services that are ranked lower then s in the rank list R_i. The cumulative score for the candidate s is:

$$B(s) = \sum_{i=1}^{n} B_i(s) \tag{4}$$

An important advantage of *Borda count* method and of other positional rank aggregation methods is that they are computationally very easy as they can be implemented in linear time. However such methods, including Borda count method, do not satisfy the Condorcet criterion [27], namely if the Condorcet winner exists is not guaranteed that it will be selected by the voting system. The Condorcet winner is that candidate that defeats any of the other candidates in two-candidates election.

Markov chains based methods In [7] a set of four methods based on Markov chains have been introduced to solve the problem of rank aggregation. A Markov chain [18] is a stochastic process consisting of a domain D, a set of states $\{s_1, ..., s_m\}$, an *initial distribution vector* $(p(s_1), ..., p(s_m))$ and a mxm *transition probability matrix*, where m is the total number of states. In a Markov chain the present state of the system captures all the information that could influence the future evaluation of the system (past states do not influence future states). The system moves from one state s_i to another state s_j with a probability P_{ij}. Rank aggregation methods based on Markov chains try to compute the stationary distribution of the Markov chain which represents the resulting rank aggregation list.

The Markov chains based methods proposed in [7] are using different heuristics rules to construct the transition probability matrix. In [7] the states are basically the web pages rank by rank engines, while in our case services represent states of a Markov chain. Given a service S_i, representing a current given state of the system, the four heuristics introduced in [7], denoted shortly by MC1, MC2, MC3 and MC4 are informally defined as follows:

- **MC1**: Choose uniformly from the multiset of all candidate services that were ranked at least as high as S_i in a rank list R.
- **MC2**: Choose a rank list R uniformly at random and pick uniformly at random from among the candidate services that were ranked at least as high as S_i in R.
- **MC3**: Choose a rank list R uniformly at random and pick uniformly at random a service S_j. If service S_j was ranked higher then current state (service S_i) in the rank list R, then select S_j as current state, otherwise stay in S_i.
- **MC4**: Choose a candidate service S_j at random. If S_j was ranked higher than S_i in most rank lists then choose S_j as current state, otherwise stay in S_i.

As pointed out in [7], method MC4 outperforms the other methods as well as the Borda count methods. We investigate all the methods introduced above on a Semantic Web services dataset and we report results in Section 6.

4 Describing and ranking Semantic Web services

This section briefly introduces our approach for modeling non-functional properties of Semantic Web services and the ranking algorithm we used for ranking services based on non-functional properties. As a model framework and language to semantically describe services we use the Web Service Modeling Ontology (WSMO) [22], respectively the Web Modeling Language (WSML) [3]. WSMO/L is a comprehensive approach for modeling Semantic Web services that offers complete rule-based modeling support required in our scenarios.

Non-functional properties of services are modeled by means of logical rules in which the terminology (e.g. concepts, relations) is provided by a set of non-functional properties ontologies [1]. Listing 4 displays a concrete example on how to describe the *obligations* non-functional property of a service.

```
namespace {_"WS1.wsml#",
   so _"Shipment.wsml#",
   wsml _"http://www.wsmo.org/wsml/wsml−syntax/"}

webService ws1Service
  nonFunctionalProperty deliveryTime hasValue ?deliveryTime
    definedBy
    //delivery time per order/package depends on dimension of the package; WS1 deliveryTime are as follows:
    // 1 day if volume <= 20 cm3, 3 days if volume > 20 cm3
    computeDeliveryTime(?package, ?deliveryTime):? ?package[so#length hasValue ?length, so#width hasValue ?
           width,
      so#height hasValue ?height] memberOf so#Package and ?volume = (?length ? ?width ? ?height) and ?
             volume < 20 and ?deliveryTime = 1.
    computeDeliveryTime(?package, ?deliveryTime):? ?package[so#length hasValue ?length, so#width hasValue ?
           width,
      so#height hasValue ?height] memberOf so#Package and ?volume = (?length ? ?width ? ?height) and ?
             volume = 20 and ?deliveryTime = 1.
    computeDeliveryTime(?package, ?deliveryTime):? ?package[so#length hasValue ?length, so#width hasValue ?
           width,
      so#height hasValue ?height] memberOf so#Package and ?volume = (?length ? ?width ? ?height) and ?
             volume > 20 and ?deliveryTime = 3.
```

Listing 1: WS1 obligations

More informally the service delivery time is as follows: in case the volume of the package is lower or equal than $20cm^3$ then the delivery time is one day, otherwise if the volume of the package is more than $20cm^3$ the delivery time is three days.

The ranking algorithm that consider multiple non-functional properties of Semantic Web services such as the one presented in the previous example was introduced as part of our previous work [25]. In rest of this section we shortly recap how the algorithm is working. A particularity of our ranking algorithm is the evaluation of the logical rules used to model non-functional properties of services by a reasoning engine. In a nutshell the multi-criteria ranking algorithm works as follows. First the non-functional properties the user is interested and their importance are extracted from the user goal description. The importance of a non-functional property is an number between 0 and 1, 0 denoting no user interest on that property, 1 denoting maximum user interest. Each service from the available set of services in the repository is checked if its description contains the non-functional properties specified in the user goal. If this is the case the corresponding logic rules are extracted and evaluated using a reasoning engine which support WSML rules. A matrix containing the services, as first dimension, and the set of non-functional properties the user is interested, as the second dimension, is constructed. The elements of this matrix are the non-functional properties values of each service obtain during the previous rule evaluation step. The matrix values are normalized and a aggregated score is computed for each service taking into account the importance of the non-functional properties. Finally the scores values are sorted and the final list of services is returned. For more details we refer the reader to our previous work [25].

[1] www.wsmo.org/ontologies/nfp

5 An approach for Semantic Web service ranking optimization

Having introduced our ranking approach for Semantic Web services (Section 4) and a set of general rank aggregation methods (Section 3), this section proposes an an optimization of the initial ranking approach for Semantic Web services by use of parallel processing and rank aggregation methods. The experimental results (Section 6) show considerable improvements in terms of *processing time* and *quality* of results.

The first step of the optimized approach is to split the set of non-functional properties requested by the user into multiple subsets. If $[NFP_1, NFP_2, ..., NFP_n]$ is the full set of non-functional properties the user is interested, the resulting subsets are of form $\Lambda_k = [NFP_1^k, ..., NFP_{|\Lambda_k|}^k]$, k is the total number of subsets. A non-functional property will be part of only one subset. More formally if $NFP \in \Lambda_k$ then $NFP \notin \Lambda_j$, where $j \neq k$. In our experiments the subsets of non-functional properties Λ, have the same cardinality.

As described in Section 4, the initial ranking algorithm constructs a matrix containing the services, as one dimension, and the set of non-functional properties the user is interested, as the other dimension. The elements of this matrix are the non-functional properties values of each service that are computed by evaluating the logical expressions (logic rules) representing the non-functional properties. Splitting the set of non-functional properties requested by the user into multiple subsets results in splitting the services/non-functional properties matrix, as exemplified in Figure 1.

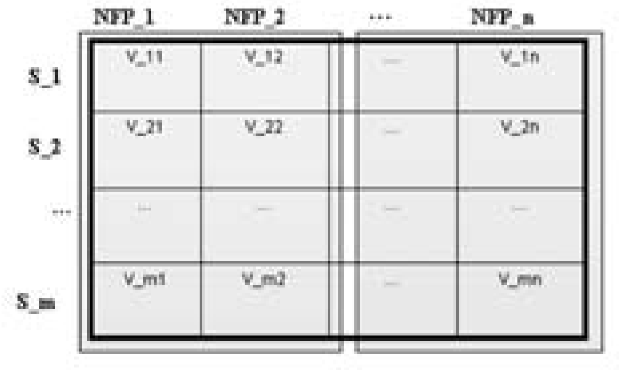

Abbildung 1: Optimization approach.

The second step is to apply the ranking algorithm described in Section 4 on each subset of non-functional properties. The ranking algorithm is applied in parallel to subsets Λ of the initial set of non-functional properties. The initial pre-processing part, including the extraction of user preferences from the goal as well as the extraction of non-functional properties descriptions from the services is performed at once for all the multiple instances of the algorithm. The evaluation of the logical expression representing the non-functional properties is done in parallel which reduces the processing time. It is known that the use of a reasoner on big data set is a time consuming task. The following steps of the initial ranking algorithm are followed for each of the subsets. This includes the normalization of the values obtained, the computation of each service score by aggregating its corresponding non-functional properties values from the given subset and finally the ordering of the services based on their scores. A rank list of services is obtained for each of the subsets of the non-functional properties set requested by the user.

The third and final step of the overall optimized Semantic Web service ranking solution includes the use of rank aggregation methods to aggregate the rank lists of services obtained as the result of the previous step. Figure 2 depicts the last step of the approach.

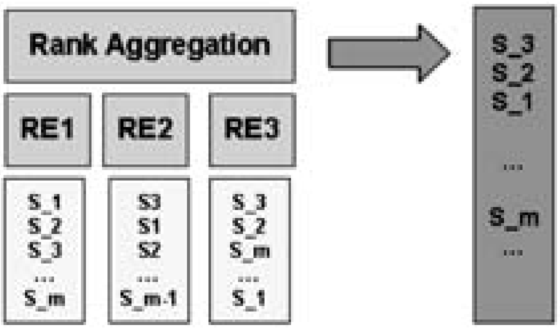

Abbildung 2: High level overview of aggregation.

Each instance of the initial ranking algorithm, depicted in the figure as RE_{Λ_i}, produces a rank list R_{Λ_i}. The rank lists are finally aggregated using rank aggregation methods introduced in Section 3.

6 Experiments, results and discussions

In this section we report the results obtained by applying first the initial, non-optimized method for ranking Semantic Web services and second a set of rank aggregation methods as part of the optimized approach.

6.1 Data set and experimental setup

In our experiment we use a set of 50 Semantic Web services each having 4 non-functional properties descriptions. The non-functional properties that we consider are: *price and discounts*, *obligations*, *delivery time* and *rewards*. *Price and discounts* represent the price charged by the shipping service to ship a package from one place to another. On top of it some shipping services might provide discounts if certain conditions are fulfilled. As obligations we model any payment obligations of the enterprise providing the shipping service in case the package to be delivered is lost or destroyed. The *delivery time* is simply the required time expected by the provider to deliver the package. Finally as *rewards* we modeled reward points that shipping service might offer to their customers.

The set of Semantic Web services used in our experiments was modeled starting from concrete shipment services descriptions. The WSDLs of these services were provided to us by seekda[2]. The set includes as well the 5 shipment services from the SWS Challenge Shipment Discovery scenario[3]. The initial WS Challenge Shipment Discovery scenario have been extended the by augmenting services description with the previously mentioned 4 non-functional properties. Similarly non-functional properties of WSDL services from the seekda collection are semantically described using the WSMO approach.

6.2 Results

Table 1 contains the average execution times in milliseconds for the initial ranking algorithm and the five rank aggregation methods (i.e., Borda, MC1, MC2, MC3 and MC4). For each algorithm/method, the execution time reported is the arithmetic mean of the execution times of 10 trials. For each of the five rank aggregation methods we consider two cases, one in which two ranking lists are being aggregated and the second in which three ranking lists are being aggregated. The user request used in all cases is the WSML goal that corresponds to informal query: "Rank the services based on the combination of price/discounts, liability/obligations, delivery time and rewards, where price/discounts and liability/obligations are equally important, delivery time and rewards are equally important but the first two are more important than the second two. The results must be in descending order."

The resulting execution times are available in Table 1.

	1 ranking list (in millisec)	**2 ranking lists (in millisec)**	**3 ranking lists (in millisec)**
Initial	197362	-	-
Borda	-	155248	145915
MC1	-	155642	145883
MC2	-	153344	144757
MC3	-	153735	145612
MC4	-	154889	144738

Tabelle 1: Ranking framework - Average execution times.

As reported in Table 1, the optimized approached based on parallelization and rank aggregation techniques performs better in terms of execution time than the initial ranking approach. Besides execution time our evaluation focuses also on qualitative performance. The qualitative performance results for each method is quantified based on the ranking metrics defined in Section .

Table 2 contains the Kendall's τ correlation values between the ranking lists results generated using the optimized approaches and the reference ranking list for Query Q_5.

Table 3 contains the Spearman's ρ correlation values between the ranking lists results

[2]seekda.com

[3] http://sws-challenge.org/wiki/index.php/Scenario:_Shipment_Discovery

	1	2	3
Initial	τ=0.81	-	-
Borda	-	τ=0.87	τ=0.81
MC1	-	τ=0.82	τ=0.80
MC2	-	τ=0.73	τ=0.76
MC3	-	τ=0.79	τ=0.73
MC4	-	τ=0.92	τ=0.81

Tabelle 2: Ranking framework - Qualitative performance Kendall's τ.

generated using the optimized approaches and the reference ranking list for Query Q_5.

	1	2	3
Initial	ρ=0.95	-	-
Borda	-	ρ=0.92	ρ=0.90
MC1	-	ρ=0.86	ρ=0.84
MC2	-	ρ=0.88	ρ=0.91
MC3	-	ρ=0.90	ρ=0.88
MC4	-	ρ=0.98	ρ=0.93

Tabelle 3: Ranking framework - Qualitative performance Spearman's ρ.

6.3 Discussions

The execution times available in Table 1 show that the rank aggregation methods perform better than the initial ranking algorithm. The percent of improvement ranges between 21% to 26%. One can observe a gradual improvement of the results by splitting the set of non-functional properties further. Rank aggregation methods applied on three ranking lists performs better than the rank aggregation methods applied on two ranking lists, where each list is determined by one or two of the four non-functional properties mentioned before.

For the qualitative evaluation we use two ranking metrics, namely Kendall's τ and Spearman's ρ to measure the quality of each approach. More precisely, we measure the distance between the ranking lists produced by the ranking approaches and the reference rank list created by human experts. A Kendall's τ and a Spearman's ρ value closer to 1 indicates that the ranking list produced by the methods and the reference rank list are highly correlated and thus the method performs well. By contrast, a Kendall's τ and a Spearman's ρ value closer to -1 indicates a higher disagreement between the two lists. One can notice that the approaches based on rank aggregation methods perform in general better than the initial approach. Among the rank aggregation methods, the MC4 method produces the best results.

7 Related work

To the best of our knowledge *rank aggregation* has not been investigated in the context of aggregating rank lists of Semantic Web services nor rank lists of Web services. However, the rank aggregation problem has been studied in the context of social choice theory, statistics, machine learning, web search engines as well as sports and competitions. The pioneering work for developing rank aggregation methods was motivated by challenges from the social theory. One of the most common rank aggregation developed in this context is Borda count [1], shortly described in Section 3. In short the method orders the candidates with respect to the average rank. In [7], rank aggregation methods were discussed in the context of the Web. The rank aggregation called *Kemeny optimal aggregation* is introduced as the optimal rank aggregation solution which optimizes the Kendall distance [12]. In [6], the same authors provide a detailed theoretical analysis of various rank aggregation methods showing that the *Kemeny optimal aggregation* is a NP-hard problem. The authors propose a set of methods based on Markov chains are proposed that approximate the Kemeny optimal. In [16] user preferences are expressed by means of ontologies and rank aggregation methods are used to combine rank lists from various attributes of user preferences when searching the Web. A detail comparison of various rank aggregation methods, including Markov chains based methods, is performed in [20] using TREC data. Furthermore the authors of this work distinguish between rank based and score based rank aggregation methods. Machine learning techniques have been used to solved the rank aggregation problem using either unsupervised learning (e.g. [14]) or supervised learning(e.g. [17]). However these approaches require good training data which is often not easy to obtain.

8 Conclusions

In this paper we have developed an optimization technique for ranking Semantic Web services using rank aggregation techniques. The initial ranking algorithm evaluates the non-functional properties specifications of each service and computes an aggregated score. Our optimized approach uses a parallel version of the initial ranking algorithm and as a last step applies rank aggregation methods to create the final aggregated rank list. We have proposed and test several rank aggregation methods including Borda count and Markov chain based methods. As future work we plan to investigate other rank aggregation methods in the context of rank aggregation for Semantic web services, including both rank and score based methods. We plan as well to perform an extended evaluation of our approaches using a larger data set, i.e. more than 50 services, with a larger number of non-functional descriptions. Another related research direction that we plan to investigate is the use of similarity measures when applying rank aggregation methods for Semantic Web services.

Literatur

[1] J. C. Borda. Memoire sur les elections au scrutin. *Histoire de l'Academie Royale des Sciences*, 1781.

[2] L. Chung. Non-Functional Requirements for Information Systems Design. In *Proceedings of the 3rd International Conference on Advanced Information Systems Engineering - CAiSE'91, April 7-11, 1991 Trodheim, Norway*, LNCS, pages 5–30. Springer-Verlag, 1991.

[3] J. de Bruijn, H. Lausen, R. Krummenacher, A. Polleres, L. Predoiu, M. Kifer, D. Fensel, I. Toma, N. Steinmetz, and M. Kerrigan. The Web Service Modeling Language WSML. Technical report, WSML, 2007. WSML Final Draft D16.1v0.3. http://www.wsmo.org/TR/d16/d16.1/v0.3/.

[4] Robert DeConde, Sarah Hawley, Seth Falcon, Nigel Clegg, Beatrice Knudsen, and Ruth Etzioni. Combining results of microarray experiments: A rank aggregation approach. *Statistical Applications in Genetics and Molecular Biology*, 5(1):15, 2007.

[5] P. Diaconis. *Group Representations in Probability and Statistics*, volume 11 of *Lecture Notes — Monograph series*. Institute of Mathematical Statistics, Hayward, CA, 1988.

[6] Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank aggregation revisited. Technical report.

[7] Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank aggregation methods for the web. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 613–622, New York, NY, USA, 2001. ACM.

[8] Dieter Fensel and Christoph Bussler. The Web Service Modeling Framework (WSMF). *Electronic Commerce Research and Applications*, 1(2):113–137, 2002.

[9] Dieter Fensel, James A. Hendler, Henry Lieberman, and Wolfgang Wahlster, editors. *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential [outcome of a Dagstuhl seminar]*. MIT Press, 2003.

[10] C. A. Tovey J. J. Bartholdi and M. A. Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, 6(2):157165, 1989.

[11] James P. Keener. The perron-frobenius theorem and the ranking of football teams. *SIAM Rev.*, 35(1):80–93, 1993.

[12] Maurice G. Kendall. Rank corellation methods. *Hafner Publishing Co.*, 1955.

[13] Alexandre Klementiev, Dan Roth, and Kevin Small. An unsupervised learning algorithm for rank aggregation. pages 616–623. 2007.

[14] Alexandre Klementiev, Dan Roth, and Kevin Small. Unsupervised rank aggregation with distance-based models. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 472–479, New York, NY, USA, 2008. ACM.

[15] Steffen Lamparter, Anupriya Ankolekar, Rudi Studer, and Stephan Grimm. Preference-based selection of highly configurable web services. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 1013–1022, New York, NY, USA, 2007. ACM.

[16] Lin Li, Zhenglu Yang, and Masaru Kitsuregawa. Using ontology-based user preferences to aggregate rank lists in web search. In Takashi Washio, Einoshin Suzuki, Kai Ming Ting, and Akihiro Inokuchi, editors, *PAKDD*, volume 5012 of *Lecture Notes in Computer Science*, pages 923–931. Springer, 2008.

[17] Yu-Ting Liu, Tie-Yan Liu, Tao Qin, Zhi-Ming Ma, and Hang Li. Supervised rank aggregation. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 481–490, New York, NY, USA, 2007. ACM.

[18] S. P. Meyn and R. L. Tweedie. *Markov chains and stochastic stability.* Springer–Verlag, 1993.

[19] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[20] Elena M. Renda and Umberto Straccia. Web metasearch: Rank vs. score based rank aggregation methods.

[21] M. Elena Renda and Umberto Straccia. Web metasearch: rank vs. score based rank aggregation methods. In *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*, pages 841–846, New York, NY, USA, 2003. ACM.

[22] D. Roman, H. Lausen, and U. Keller (Ed.). Web service modeling ontology (WSMO). Working Draft D2v1.4, WSMO, 2007. Available from http://www.wsmo.org/TR/d2/v1.4/.

[23] C. Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 100(3/4):441–471, 1987.

[24] M. Stob. A supplement to a mathematicians guide to popular sports. *American Mathematical Monthly*, 91(5):277–282, 1984.

[25] Ioan Toma, Dumitru Roman, Dieter Fensel, Brahmanada Sapkota, and Juan Miguel Gomez. A multi-criteria service ranking approach based on non-functional properties rules evaluation. In *ICSOC '07: Proceedings of the 5th international conference on Service-Oriented Computing*, pages 435–441, Berlin, Heidelberg, 2007. Springer-Verlag.

[26] Le-Hung Vu, Manfred Hauswirth, and Karl Aberer. Qos-based service selection and ranking with trust and reputation management. In *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, volume 3760/2005, 2005.

[27] H. P. Young. Condorcet's theory of voting. *The American Political Science Review*, 82(4):1231–1244, 1988.