Extending Datalog for Matchmaking in P2P E-Marketplaces

Azzurra Ragone, Umberto Straccia, Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini

1 Politecnico di Bari, Via Re David, 200, I-70125, Bari, Italy
{a.ragone, t.dinoia, disciascio}@poliba.it
2 ISTI - CNR, Via G. Moruzzi 1, 56124 Pisa, Italy
straccia@isti.cnr.it
3 Università della Tuscia, via San Carlo, 32, I-01100, Viterbo, Italy
donini@unitus.it

Abstract. We present an approach to matchmaking in P2P e-marketplaces, which mixes in a formal and principled way Datalog, fuzzy sets and utility theory, in order to determine most promising matches between perspective counterparts. Use of Datalog ensures the scalability of our approach to large marketplaces, while Fuzzy Logic provides a neat connection with logical specifications and allows to model soft constraints and how well they could be satisfied by an agreement. Note-worthy is that our approach takes into account in the peer-to-peer matchmaking also preferences of each counterpart and their utilities. This allows to rule out of the match list those counteroffers that, although seemingly appealing for the buyer, would probably lead to failure due to contrasting preferences of the seller, and paves the way to the actual negotiation stage.

1 Introduction

In e-commerce settings, matchmaking can be defined as the process of finding “good” counterparts for a given entry in an marketplace. While in e-marketplaces dealing with undifferentiate products (as commodities) the “best” counterpart is the one offering the “best” price per unit, in e-marketplace dealing with complex products —cars, houses, computers— the price is not the single feature to take into account. Also, ads can involve bundle of issues, e.g., Sports car with optional package including both GPS system and alarm system or implications e.g., If a car has leather seats then it is also provided with air conditioning, and some kind of logical theory, able to let users express their needs/offers, could surely help. A matchmaking system has to be able both to find “good” counterparts and to evaluate how “good” a counterpart is. Currently, many commercial sites force the buyer to enter her request browsing a predefined classification that may be completely unsuitable for the characteristics the buyer might have in mind e.g., they require to enter a brand first, then a model of that brand, etc. while a buyer may be not interested in a specific brand, but only on some limitations on price and color. In this respect, one may say that they provide no matchmaking assistance: the matchmaker is the buyer herself.

To assist buyers and sellers in marketplaces, several research proposals on matchmaking systems were issued. They either try to compute a score of possible counterparts, based on textual information [24], or to compare the logical representations of supply and demand [9], or combine both scores and logic in some way [6, 14]. Our proposal falls in this last category, mixing in a formal way Datalog, Fuzzy sets, and Utility Theory. While the above
logic-based proposals do not tackle the scalability problem, our resort on Datalog ensures the scalability of our approach to large marketplaces. On the other hand, the resort on Fuzzy Logic ensures a neat connection with the logical specification, while allowing the system to give an explanation of suggestions in terms of how well the preferences could be satisfied by an agreement.

Since we want both buyer and seller equally satisfied, the matchmaker computes a score as the maximum value of the product of the utilities of the buyer \( u_B \) and the seller \( u_S \) over all possible agreements. In this way both buyer’s and seller’s preferences are taken into account ruling out of the match list those counteroffers that, although seemingly appealing for the buyer, would probably lead to failure due to contrasting preferences of the seller. The remaining of the paper is as follow: to set the stage we introduce basics of Datalog \( \text{topk} \) and requirements for matchmaking process. Then fuzzy soft constraints are presented followed by the description of fuzzy matchmaking process in Datalog \( \text{topk} \). Discussion about relevant related work and conclusions close the paper.

2 Top-k Datalog

We use an extension of Datalog (cf. [1, 5, 23]) as our representation and query tool. We extend it by allowing soft constraint predicates to appear in rules and queries (we call the language Datalog \( \text{topk} \)). The proposal here extends the work [20] in which soft constraint predicates, i.e., fuzzy predicates, may appear in a query rules only and is conceptually equivalent to [21]. Basically, we allow vague/fuzzy predicates to occur in rule bodies, which have the effect that each tuple in the answer set of a query has now a score in \([0, 1]\). Datalog \( \text{topk} \) addresses the problem to compute the top-k answers in case the set of facts is huge, without evaluating all the tuples’ score. As matching a buyer’s request with a seller’s offer is a matter of degree, we will use Datalog \( \text{topk} \) to find the top-k matchings. Datalog \( \text{topk} \) is as Datalog except that we additionally allow fuzzy predicates to occur in Datalog rule bodies. Specifically, let \( r \) be an \( n + 1 \)-ary predicate. A Datalog \( \text{topk rule} \) is of the form

\[
\begin{align*}
\text{body}(x, y), s = f(p_1(z_1), \ldots, p_n(z_n))
\end{align*}
\]

where

1. \( x \) are the \( n \) distinguished variables;
2. \( s \) is the score variable, taking values in \([0, 1]\), and \( r \) is functional on \( s \);
3. \( y \) are so-called non-distinguished variables and are distinct from the variables in \( x \);
4. \( \text{body}(x, y) \) is a conjunction of Datalog atoms;
5. \( z_i \) are tuples of constants or variables in \( x \) or \( y \);
6. \( p_i \) is an \( n_i \)-ary fuzzy predicate assigning to each \( n_i \)-ary tuple \( c_i \) as score \( p_i(c_i) \in [0, 1] \);
7. \( f \) is a scoring function \( f: [0, 1]^n \rightarrow [0, 1] \), which combines the scores of the \( n \) fuzzy predicates \( p_i \) into and overall query score to be assigned to the score variable \( s \). We assume that \( f \) is monotone, i.e., for each \( v, v' \in [0, 1]^n \) such that \( v \leq v' \), \( f(v) \leq f(v') \) holds, where \( (v_1, \ldots, v_n) \leq (v'_1, \ldots, v'_n) \) iff \( v_i \leq v'_i \) for all \( i \);
8. We assume that the computational cost of \( f \) and all fuzzy predicates \( p_i \) is bounded by a constant.

We call \( s = f(p_1(z_1), \ldots, p_n(z_n)) \) a scoring atom. For instance,

\[
\begin{align*}
\text{CheapCar}(x, p, s) & \equiv \text{NewCar}(x), \text{CarPrice}(x, p), s = \max(0, 1 - p/15000) \\
\text{CheapCar}(x, p, s) & \equiv \text{SecondHandCar}(x), \text{CarPrice}(x, p), s = \max(0, 1 - p/7500)
\end{align*}
\]
are two Datalog $^{\text{topk}}$ rules looking for cheap cars, assigning to each car a score depending on its price. If the price of a new car is above 15000 the car is not considered as a cheap one, while the scoring function is increasing as the price lowers. Hence, it is quite natural that if we are looking for cheap cars one wants that the retrieved cars are sorted in decreasing order with respect its score, i.e., degree of cheapness. Furthermore, as the database may contain thousands of tuples, one usually wants to retrieve just the top-$k$ ranked ones.

A Datalog $^{\text{topk}}$ query program is a pair $\mathcal{P} = (\mathcal{P}', q)$, where $\mathcal{P}'$ is Datalog $^{\text{topk}}$ program, $q$ is a query rule and $q$ does not occur in $\mathcal{P}'$. Essentially, the difference between Datalog and Datalog $^{\text{topk}}$ is that now scoring atoms may appear in the rule body of “query rules”.

The basic inference services that concerns us is the top-$k$ retrieval problem, where this latter is defined as:

**Top-$k$ retrieval:** Given a Datalog $^{\text{topk}}$ program $\mathcal{P}$, retrieve the top-$k$ ranked tuples $\langle c, v \rangle$ that instantiate the query $q$ and rank them in decreasing order w.r.t. the score $v$, i.e., find the top-$k$ ranked tuples of the answer set of $q$, denoted

$$\text{ans}_k(\mathcal{P}, q) = \text{Top}_k \{ \langle c, v \rangle \mid \mathcal{P} \models q(c, v) \} .$$

For instance,

$$q(x, p, s) \leftarrow \text{CheapCar}(x, p, s)$$

is a query asking for cheap cars. The top-$k$ ranked cars, according to the score (that depends on their price), is obtained by $\text{ans}_k(\mathcal{P}, q)$.

### 3 Matchmaking Requirements

In the whole paper we adopt as reference in examples a P2P automobile marketplace, and motivate our work in this domain. With respect to this e-marketplace the users can express different preferences on several features, i.e., a buyer can specify conditional preferences, such as *if the car is a sports one, than the feeding has to be gasoline*, or *a cheap car*, yet if *the car is provided with a GPS system she is ready to pay up to 17000*.

Looking at the previous example, we notice the buyer’s requests can be split into two different parts. In fact, the buyer expresses as a hard requirement that if the car is a sports one then the feeding has to be gasoline and as a preference, soft constraint, that she is willing to buy a cheap car but she could spend some more if the car is equipped with a GPS system. Strict requirements represent what the buyer and the seller want to be necessarily satisfied in order to accept the final agreement – in our framework we call strict requirements *hard constraints*. Preferences denote issues they are willing to negotiate on – this is what we call *soft constraints*. Let us now introduce an example request, we will use to explain some aspects of our approach:

**Example 1.** Suppose to have a buyer’s request like “I want a station wagon black or gray. Preferably I would like to pay less than 14,000 € furthermore I’m willing to pay up to 17,000 € if warranty is greater or equal than 100000 km. (I don’t want to pay more than 19,000 € and I don’t want a car with a warranty less than 60,000 km)”.

In this example we identify:

- **hard constraints** = *Body Type*: Station wagon. *Color*: Black or Grey. *Price*: $\leq 19,000$ €. *Warranty*: $\geq 60,000$ km.

- **soft constraints** = *Price*: $\leq 14,000$. *Warranty-Price*: if Warranty $\geq 100,000$ then Price $\leq 17,000$ €
3.1 Preferences and Utilities

In a matchmaking process, retrieving supplies matching the request taking into account only hard constraints would be trivial. The matchmaker should just look for agreements where hard constraints of both the buyer and seller are satisfied. Given a request and a set of retrieved supplies, how should the matchmaker find—and rank—the most suitable or promising agreements to propose to both parties? The matchmaker should exploit soft constraints expressed both by the buyer and the seller.

Among the supplies completely satisfying all the requirements modeled in hard constraints, the top-ranked ones will be those best satisfying features expressed in the soft constraints proposed both by the buyer and the seller.

For what concerns soft constraints, we observe that the buyer’s and seller’s satisfaction degree, with respect to a final agreement, depends on which parts of her preference specifications have been satisfied. For instance, w.r.t. Example 1 suppose to have the following two supplies, where only soft constraints have been specified:

\( \sigma' : \text{Body Type: Station wagon. Color: Grey. Price: 16,000 } \mathcal{E}. \text{ Warranty: 200,000 km.} \)

\( \sigma'' : \text{Body Type: Station wagon. Color: Black. Price: 13,000 } \mathcal{E}. \text{ Warranty: 50,000 km} \)

Comparing these supplies with buyer’s soft constraints we note that \( \sigma' \) satisfies the preference \( \beta_2 = \{ \text{Warranty-Price: if Warranty } \geq 100,000 \text{ then Price } \leq 17,000 \mathcal{E} \} \), while \( \sigma'' \) the preference \( \beta_1 = \{ \text{Price: } \leq 14,000 \} \). How to evaluate the best one? We expect the buyer assigns a positive utility value representing the preference relevance to sub-parts of soft constraints. In this case we assume utility values — \( u(\beta_1) \) and \( u(\beta_2) \) — both for \( \beta_1 \) and \( \beta_2 \). Actually, the same holds from the seller’s perspective. In fact, in a P2P e-marketplace the seller may express his preferences — soft constraints e.g., on selling price, warranty, delivery time — with corresponding utilities \( u(\sigma_j) \), as well as his hard constraints (e.g., color, model, engine fuel, etc.).

The only constraint on utility values is that both seller’s and buyer’s ones are normalized to 1 to eliminate outliers, and make them comparable [11].

\[
\sum u(\beta_i) = 1 \quad \sum u(\sigma_j) = 1
\]

(1)

Since we assume utilities on preferences as additive, here we can write the global utility of the buyer \( u_\beta \) and of the seller \( u_\sigma \) as just a sum of the utilities of preferences satisfied in the agreement. If we are able, for each preference \( \beta_i \) of the buyer and \( \sigma_j \) of the seller, to evaluate a score \( s_i \) and \( s_j \), representing a degree of preference satisfaction, then it is reasonable to think that the global utility has to take into account also these information. Hence, in formulas, we write the two utility functions as:

\[
u_\beta = \sum s_i \ast u(\beta_i) \quad u_\sigma = \sum s_j \ast u(\sigma_j)
\]

(2)

From Example 1, we note that while considering numerical features, it is still possible to express hard and soft constraints on them. A hard constraint on a numerical feature can be considered as a reservation value [18] on the feature itself.

3.2 The Matchmaking Process

Based on the notions of preferences, utility, and reservation values introduced so far, we begin outlining the actual matchmaking process. Every time a seller (buyer) enters the marketplace, he proposes his supply (request) expressing both hard constraints and soft constraints (preferences). Then, he sets his reservation value \( r_{\sigma, f} \) on numerical feature \( f \). Eventually,
for each preference $\sigma_j$ ($\beta_i$) he expresses the corresponding utility $u(\sigma_j)$ ($u(\beta_i)$). Based on buyer’s and seller’s specifications, the matchmaker returns a ranked list of supplies such that: [a] they satisfy both the hard constraints in the request and conversely their hard constraints are satisfied by the request; [b] the rank is evaluated taking into account preferences and utility functions $u_\beta$ and $u_\sigma$ as defined by equations (2). In a P2P e-marketplace, usually the aim is to find agreements maximizing not only the buyer’s satisfaction or the seller’s one, but trying to make them equally satisfied. So the matchmaker has to propose agreements mutually beneficial for both of them. Such agreements are computed considering the higher values of $u_\beta$ and $u_\sigma$ utilities product [16].

4 Requirements Fuzzy Representation

Marketplaces are typical scenarios where the notion of fuzziness is often involved. The concept of Cheap or Expensive are quite usual. Similarly, numerical variables involved in a commercial transaction expose a fuzzy behavior. For instance, suppose to have a buyer looking for a car provided with a warranty greater than 100,000 kilometers and a supplier selling his car with a 80,000 kilometers warranty. We can not say they do not match at all. Instead we can say they match with a certain degree. A logical language able to deal with fuzzy information would be then a good choice to model matchmaking. To represent buyer/seller requirements in a Datalog topk setting, hereafter we will use the following notation:

$$\text{hard constraints} = \begin{cases} \beta(x, y) \text{ or } \beta(x), \text{ buyer’s strict requirements} \\ \sigma(x, y) \text{ or } \sigma(x), \text{ seller’s strict requirements} \end{cases}$$

$$\text{soft constraints} = \begin{cases} \beta_i(x, y, s) \text{ or } \beta_i(x, s), \text{ buyer’s preferences} \\ \sigma_j(x, y, s) \text{ or } \sigma_j(x, s), \text{ seller’s preferences} \end{cases}$$

where $x$ is a single variable, $y$ (if present) represents a vector of numerical variables and $s$ represents the score variable as defined in Section 2. Since a score is associated to each fuzzy predicate, we can compute the global utility based on the two utility functions in Section 3.1. The two predicates $\sigma$ and $\beta$ model the minimal requirements the buyer and the seller want to be satisfied in order to accept the final agreement. Notice that, if seller and buyer set hard constraints in conflict with each other, the corresponding supply will not be retrieved and no agreement will be reached. Soft constraints are modeled via Datalog predicates $\beta_i$ for the buyer and $\sigma_j$ for the seller, where each of them represents a sub-part of the buyer/seller preferences.
The use of \( x, y \) and \( s \) should be clearer looking at how buyer’s request in Example 1 is formalized:

\[
\beta_4(x) \leftarrow \text{StationWagon}(x)
\]
\[
\beta_5(x, p) \leftarrow \text{CarPrice}(x, p), 0 \leq p \leq 19000
\]
\[
\beta_6(x, \text{kmw}) \leftarrow \text{KmWarranty}(x, \text{kmw}), \text{kmw} \geq 60000
\]
\[
\beta_7(x) \leftarrow \text{Grey}(x)
\]
\[
\beta_8(x) \leftarrow \text{Black}(x)
\]
\[
\beta_9(x, p, \text{kmw}) \leftarrow \beta_4(x), \beta_5(x, p), \beta_6(x, \text{kmw}), \beta_7(x)
\]
\[
\beta_1(x, p, s) \leftarrow \text{CarPrice}(x, p),
\]
\[
\left\langle \text{LS}(0, 100000, 14000, 16000, p, s) \right\rangle
\]
\[
\beta_2(x, p, \text{kmw}, s) \leftarrow \text{KmWarranty}(x, \text{kmw}), \text{CarPrice}(x, p)
\]
\[
\left\langle \text{RS}(0, 400000, 80000, 100000, \text{kmw}, s_1), \right.\]
\[
\text{LS}(0, 100000, 17000, 19000, p, s_2),
\]
\[
\left. s = \max(1 - s_1, s_2) \right\rangle
\]

5 Fuzzy Matchmaking in Datalog \( \text{topk} \)

Now we have all what is needed to model the matchmaking framework in a Datalog \( \text{topk} \) setting. First of all we show how to write the corresponding Datalog \( \text{topk} \) program \( P_{\text{match}} = \langle P_I, P_E \rangle \).

1. for each supply write the corresponding Datalog fact and add it to the Extensional Database \( P_E \);
2. encode buyer’s hard constraints requirements as a Datalog rule where the head contains the predicate \( \beta(x, y) \) as shown in Section 4; add the rule to the Intensional Database \( P_I \);
3. encode seller’s hard constraints requirements as a Datalog rule where the head contains the predicate \( \sigma(x, y) \); add the rule to \( P_I \);
4. for each buyer’s preference \( \beta_i \), write the corresponding rule in Datalog \( \text{topk} \) where the head contains the predicate \( \beta_i(x, y, s) \) as shown in Section 4; add the rule to \( P_I \); set the utility value \( u(\beta_i) \) as shown in Section 3.1;
5. for each seller’s preference \( \sigma_j \), write the corresponding rule in Datalog \( \text{topk} \) where the head contains the predicate \( \sigma_j(x, y, s) \) as shown in Section 4; add the rule to \( P_I \); set the utility value \( u(\sigma_j) \) as shown in Section 3.1;
6. add to \( P_I \) the rules:
   \[
   \text{Buyer}(x, y, u_\beta) \leftarrow \beta(x, y_0, \beta_1(x, y_1, s_1), \beta_2(x, y_2, s_2), \ldots),
   \]
   \[
   u_\beta = u(\beta_1) \cdot s_1 + u(\beta_2) \cdot s_2 + \ldots \tag{3}
   \]
   \[
   \text{Seller}(x, y, u_\sigma) \leftarrow \sigma(x, y_0, \sigma_1(x, y_1, s_1), \sigma_2(x, y_2, s_2), \ldots),
   \]
   \[
   u_\sigma = u(\sigma_1) \cdot s_1 + u(\sigma_2) \cdot s_2 + \ldots \tag{4}
   \]
   where for each variable in \( y \) in the head of one of the two previous rules, the same variable occurs in at least one of the arrays of the corresponding body: \( y_0, y_1, y_2, \ldots \);
7. add to \( P_I \) the “query rule”:
   \[
   \text{Match}(x, y, u) \leftarrow \text{Buyer}(x, y_\beta, u_\beta), \text{Seller}(x, y_\sigma, u_\sigma),
   \]
   \[
   u = u_\beta \cdot u_\sigma \tag{5}
   \]
   where for each variable in \( y \) in the head of the rule, the same variable occurs in \( y_\beta \) or \( y_\sigma \).
Once we have the Datalog \( \text{topk} \) program \( P_{\text{match}} \), then solve the following \textbf{Top-k retrieval} problem:

\[
\text{ans}_k(P_{\text{match}}, \text{Match}) = \text{Top}_k \{ \langle x, y, u \rangle \mid \langle y, u \rangle \in \text{Top}_1 \{ \langle x, y', u' \rangle \mid P \models \text{Match}(x, y', u') \} \}.
\]

Basically, for each key value \( x \) of the database, we compute the best matching \( \langle y, u \rangle \) for it, \( i.e., \langle y, u \rangle \in \text{Top}_1 \{ \langle x, y', u' \rangle \mid P \models \text{Match}(x, y', u') \} \), and then rank the top-\( k \) key values.

Notice that the rank is computed considering the product of buyer’s and seller’s utilities as stated at the end of Section 3.2.

### 6 Related Work and Discussion

In this paper we presented a fuzzy matchmaking approach exploiting Datalog to find the most promising agreements in a P2P e-marketplace. More precisely, exploiting fuzzy rules we have been able to model \textit{soft constraints}, while taking into account both buyer’s and seller’s preferences and utilities to find agreements mutually beneficial for them both. The P2P matchmaking process is modeled as a Datalog program able also to consider background domain knowledge while keeping the approach effective and scalable. A prototype is currently being implemented to further validate the approach through large scale experiments. Recently, the problem of matchmaking has been investigated under different perspectives and many approaches have been proposed. An initial approach to matchmaking can be dated back to vague query answering [15] where the need to go beyond pure relational databases was addressed using weights attributed to several search variables. More recently similar approaches have been proposed extending SQL with “preference” clauses, in order to allow relaxed queries in structured databases [10] where only buyer’s preferences are taken into account while retrieving promising supplies. Classified-ads matchmaking, at a syntactic level, was proposed in [19] and [24] to perform a matchmaking between semi-structured descriptions. Approaches to matchmaking using LOOM as description language can be found, among others, in [3] and [8]. Due to the growing interest in the Semantic Web initiative many approaches to matchmaking have been proposed based on Description Logics (DL). Matchmaking as satisfiability of concept conjunction in DLs was first proposed in [9]. In the framework of Retsina Multiagent infrastructure [22], a specific language was defined for agent advertisement, and matchmaking engine was developed [17], which carries out the process on five possible match levels. This approach was later extended in [13], with two new levels for matching classification. A similar classification was proposed — in the same venue — in [7], along with properties that a matchmaker should have in a DL based framework, and algorithms to classify and semantically rank matches within classes. An initial DL-based approach, adopting penalty functions ranking, has been proposed in [4], in the framework of dating systems. An extended matchmaking approach, with negotiable and strict constraints in a DL framework has been proposed in [6], using both concept contraction and concept abduction. Approximation and ranking in DL-based approaches to matchmaking has also recently led to adopting fuzzy-DLs, as in sMART [2] or hybrid approaches, as in the OWLS-MX matchmaker [12]. In [14] a language able to express conditional preferences is proposed to perform a matchmaking in Description Logics. Also in this case nothing is said on how to compute a agreement — as needed in P2P scenarios. Furthermore, the notion of fuzzy/vague requirements is not addressed.
References