Top-k Web Service Composition in the Context of User Preferences

Karim Benouaret [1], Djamal Benslimane[1], Allel Hadjali [2]

[1]LIRIS, University of Lyon
{karim.benouaret, djamal.benslimane}@liris.cnrs.fr
[2]IRISA, Rennes1 University
allel.hadjali@enssat.fr

1er septembre 2011

Outline

Outline

1. Introduction
2. Service composition based preference queries
3. Top-k service composition
4. Experimental evaluation
5. Conclusion

Problem description

- Data Web services
  - network accessible software entities
  - returning some information to the user (e.g., a weather forecast service or a news service)
- Data Web service composition
  - a combination of primitive Data Web services
  - answering user's complex queries
- User preferences
  - important to customize the composition process
  - rank-order the Data Web service compositions
  - flexible manner : linguistic terms (e.g., “rather cheap" or "‘not expensive")
  - modeled using fuzzy sets
- Objective : find the top-k Data Web service compositions according to user preferences

Example

Service	Functionality	Constraints
$S_{11}(\$x, ?y)$	Returns the automakers y in a given country x	-
$S_{21}(\$x, ?y, ?z, ?t)$	Returns the cars y along with their prices z and warranties t for a given automaker x	z is $cheap$, t is $short$
$S_{22}(\$x, ?y, ?z, ?t)$		z is $accessible$, t is $[12, 24]$
$S_{23}(\$x, ?y, ?z, ?t)$		z is $expensive$, t is $long$
$S_{24}(\$x, ?y, ?z, ?t)$		z is $[9000, 14000]$, t is $[6, 24]$
$S_{31}(\$x, ?y, ?z)$	Returns the power y and the consumption z for a given car x	y is $weak$, z is $small$
$S_{32}(\$x, ?y, ?z)$		y is $ordinary$, z is $approximately\ 4$
$S_{33}(\$x, ?y, ?z)$		y is $powerful$, z is $high$
$S_{34}(\$x, ?y, ?z)$		y is $[60, 110]$, z is $[3.5, 5.5]$

Q_1 :"return the French cars, preferably at an affordable price with a warranty around 18 months and having a normal power with a medium consumption"

Overview of our approach

- Challenges
  - how to retain the most relevant services
  - how to generate the top-k compositions
- Contribution
  - compute matching degrees between user preferences and services' constraints
  - propose a ranking criteria based on a fuzzufication of Pareto dominance to select the most relevant services/compositions
  - to avoid returning similar compositions, we also propose a diversified top-k compositions

Outline

Terminology

- Q :-$(q_1, ..., q_n)$: a preference query
- $\mathcal{S} = \{S_1, ..., S_n\}$: a set of service classes
- $S_i = \{S_{i1}, ..., S_{in_i}\}$: a set functionally similar services
- $S_i \sqsubseteq q_i$: services of S_i can be used to answer q_i
- $\mathcal{M} = \{M_1, ..., M_m\}$ a set of matching methods

Matching degrees between services and query components

S_{ij}	q_i	CBM	G-IBM	L-IBM	K-IBM
S_{11}	q_1	-	-	-	-
S_{21}	q_2	$(1, 0.57)$	$(1, 0)$	$(1, 0)$	$(0.80, 0)$
S_{22}		$(0.89, 1)$	$(0, 1)$	$(0.90, 1)$	$(0.50, 1)$
S_{23}		$(0.20, 0.16)$	$(0, 0)$	$(0, 0)$	$(0, 0)$
S_{24}		$(0.83, 0.88)$	$(0.60, 0.50)$	$(0.60, 0.50)$	$(0.60, 0.50)$
S_{31}	q_3	$(0.50, 0.36)$	$(0, 0)$	$(0, 0)$	$(0, 0)$
S_{32}		$(0.79, 0.75)$	$(0, 0.25)$	$(0.60, 0.50)$	$(0.40, 0.50)$
S_{33}		$(0.21, 0.64)$	$(0, 0)$	$(0, 0)$	$(0, 0)$
S_{34}		$(0.83, 0.85)$	$(0.50, 0.50)$	$(0.50, 0.50)$	$(0.50, 0.50)$

Outline

Current approaches

- Scoring function
  - computes a score for each service as an aggregate of the individual matching degrees
  - requires users to assign weights to individual matching degrees
  - users lose the flexibility to select their desired services
  - one matching method
- Skyline
  - compromises the services which are not nominated
  - privileges services with a large variance
  - one matching method

Pareto dominance vs fuzzy dominance

- Pareto dominance :
  $u \succ v \iff \forall i \in [1, d], u_i \geq v_i \wedge \exists k \in [1, d], u_k > v_k$
- Fuzzy dominance : $deg(u \succ v) = \frac{\sum_{i=1}^{d} \mu_{\gg}(u_i, v_i)}{d}$, where
  $$\mu_{\gg}(x, y) = \left\{ \begin{array}{cc} 0 & if x - y \leq \varepsilon \\ 1 & if x - y \geq \lambda + \varepsilon \\ \frac{x-y-\varepsilon}{\lambda} & otherwise \end{array} \right\}$$
- Comparison $(u = (1, 0), v = (0.90, 1))$
  - neither $u \succ v$ nor $v \succ u$
  - $deg(u \succ v) = 0.25$ and $deg(v \succ u) = 0.50$ ($\varepsilon = 0, \lambda = 0.2$)

Associating score with a Service/Composition

- Service's score : $S_{ij} \in \mathcal{S}_i$, indicates the average extent to which S_{ij} dominates the whole services of its class $\mathcal{S}_i$
$FDS(S_{ij}) = \frac{1}{(|\mathcal{S}_i|-1)m^2} \sum_{\imath=1}^{m} \sum_{\substack{S_{ik} \in \mathcal{S}_i \\ k \neq i}} \sum_{\jmath=1}^{m} deg(S_{ij}^{\imath} \succ S_{ik}^{\jmath})$

- Composition's score : $C = \{S_{1j_1}, ..., S_{nj_n}\}$
$FDS(C) = \frac{1}{d} \sum_{i=1}^{n} d_i \cdot FDS(S_{ij_i})$

An efficient generation of top-k compositions

- straightforward method :
  - generate all possible compositions
  - compute their scores
  - return the top-k ones
  - high computational cost
- Optimization technique (*theorem 1*) : $C = \{S_{1j_1}, ..., S_{nj_n}\}$ $\exists S_{ij_i} \in C; S_{ij_i} \notin top\text{-}k.\mathcal{S}_i \Longrightarrow C \notin top\text{-}k.\mathcal{C}$.

An efficient generation of top-k compositions (our example)

Services	Class	Score	Top-k
S_{11}	$\mathcal{S}_1$	-	S_{11}
$\cancel{S_{21}}$	$\mathcal{S}_2$	0.487	S_{22} S_{24}
S_{22}		0.653	
$\cancel{S_{23}}$		0.035	
S_{24}		0.538	
$\cancel{S_{31}}$	$\mathcal{S}_3$	0.094	S_{32} S_{34}
S_{32}		0.593	
$\cancel{S_{33}}$		0.130	
S_{34}		0.743	

Compositions	Score	Top-k
$C_1 = \{S_{11}, S_{22}, S_{32}\}$	0.623	C_2 C_4
$C_2 = \{S_{11}, S_{22}, S_{34}\}$	0.698	
$C_3 = \{S_{11}, S_{24}, S_{32}\}$	0.566	
$C_4 = \{S_{11}, S_{24}, S_{34}\}$	0.640	

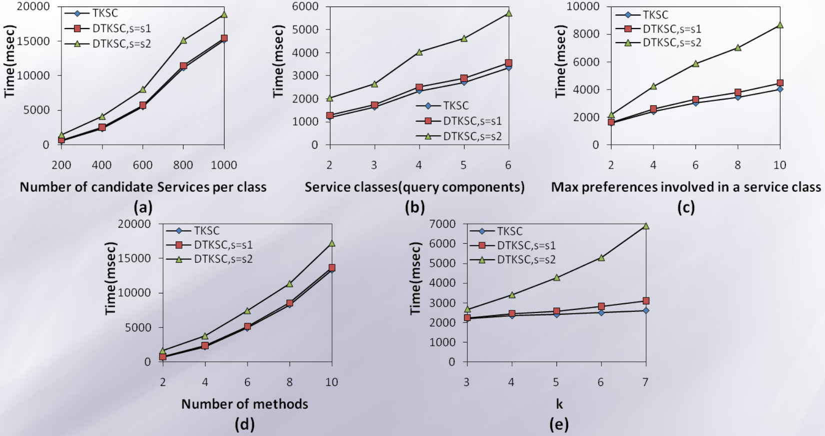
- Straightforward method : 16 compositions ($\prod_{i=1}^{n_i} |\mathcal{S}_i|$)
- Our method : 4 compositions($\leq k^{n_i}$)

Diversity-aware Top-k Compositions

- Different similar services could exist in each class S_i leading to similar compositions
- Diversification is then needed to improve user satisfaction
- $Quality(S_{ij}) = FDS(S_{ij}) \times RelDiv(S_{ij}, dtopk.\mathcal{S}_i)$
- $RelDiv(S_{ij}, dtopk.\mathcal{S}_i) =$
$$\begin{cases} 1 & dtopk.\mathcal{S}_i = \emptyset \\ \frac{\sum_{S_{ik} \in dtopk.\mathcal{S}_i} Dist(S_{ij}, S_{ik})}{|dtopk.\mathcal{S}_i|} & otherwise \end{cases}$$

Outline

time vs Parameters



Outline

Conclusion & Future work

- Conclusion
  A framework that identify and retrieve the most relevant services and return the top-k compositions according to the user preferences
- Future work
  - user study to evaluate the quality of the results
  - Combine with QoS

Q & A

Thank you