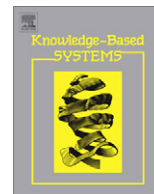




Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Combining weights with fuzziness for intelligent semantic web search

Hai Jin^{a,*}, Xiaomin Ning^a, Weijia Jia^b, Hao Wu^a, Guilin Lu^c^aServices Computing Technology and System Laboratory, Cluster and Grid Computing Laboratory, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China^bDepartment of Computer Science, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong^cDepartment of Mathematics and Statistics, Shanghai LiXin University of Commerce, Shanghai 201620, China

ARTICLE INFO

Article history:

Received 12 July 2007

Accepted 24 March 2008

Available online xxx

Keywords:

Fuzzy description logic

Intelligent search

Rank

Semantic web

User preference

Weighting

ABSTRACT

Intelligent retrieval for best satisfying users search intentions still remains a challenging problem due to the inherent complexity of real-world semantic web applications. Usually, a search request contains not only vagueness or imprecision, but also personalized information goals. This paper presents a novel approach which formulates one's search request through tightly combining fuzziness together with the user's subjective weighting importance over multiple search properties. A special ranking mechanism based on the weighed fuzzy query representation is proposed. The ranking method generates a set of "degree of relevance" – an overall score which reflects both fuzzy predicates and the user's personalized preferences in the search request. Moreover, the ranking method is general and unique rather than arbitrary. Hence, search results shall be properly ordered in terms of their relevance with respect to best matching the search intention. The experimental results show that our approach can effectively capture users information goals and produce much better search results than existing approaches.

© 2008 Elsevier B.V. All rights reserved.

1. Introduction

The semantic web is the vision of having data on the web defined and linked in a way that it can be used by machines not just for display purposes, but for automation, integration and reuse of data across various applications [4]. Extensive efforts have been made for the realization of the semantic web layered architecture. At present, the ontology layer in which data is encoded in ontology languages such as RDF/RDFS and OWL DL [3] rooted in Description Logics (DLs) [2], has reached sufficient maturity. Accordingly, some query languages which usually adopt an expressive SQL-like declarative syntax form including RQL, RDQL, SeRQL and SPARQL [37], have also been developed to accurately access the semantic web data. Hence, many recent research interests have focused on rules and proofs, such as rule languages of Description Logic Program (DLP) [14], SWRL [17] and their counterpart inference engines (e.g., FaCT++ [10], KAON2 [21], RacerPro [33]) which allow computers to conduct automated reasoning to deduce new information from knowledge bases (KBs).

Due to the intrinsic complexity of real-world knowledge, however, intelligent retrieval of semantic web data to best capture users search intentions still remains an open problem [42]. Firstly,

the information intension of a search request usually involves vagueness or imprecision which pervasively other than merely an exception exists in many real-world applications of the semantic web such as multimedia processing and retrieval [12,27], medical diagnosis [13] and geospatial [7]. Consider, for instance, a knowledge base on scientific literatures. A Ph.D. candidate may raise search requests like "find *recent* publications *written by* Zadeh" or "find *highly* cited publications *talking about* description logic". In this case, "*recent*", "*highly*" and even "*about*" are so-called fuzzy or vague predicates [43] since they can hardly have a precise definition as the predicate "*written by*" does have. Current standard semantic web ontology languages as well as rule inferences are based on bivalent logic (i.e., either *true* or *false*) and hence fail to represent the above vague knowledge. Secondly, one may have his/her own quality measurement or favoritism among multiple criteria for interesting search results. That is, the user might inequally value the multiple criteria. For instance, the Ph.D. candidate issues the following request over the literatures KB: finding *recent* publications *highly* cited *written by* Zadeh, but *prefer* more weight on recentness *over* citation. Consider another example in a multimedia semantic web application. A user might want to retrieve movie clips which have a *predominantly* green scene with a *loud* noise in the sound track, but *favor* twice as much about the color of the picture *over* as about the loudness of the sound. Here, "*prefer A over B*" or "*favor A over B*" indicates the user's own importance measurements on two different criteria (i.e., A and B). Finally, unlike classical semantic web query languages

* Corresponding author. Tel.: +86 27 8754 3529; fax: +86 27 8755 7354.

E-mail addresses: hjin@hust.edu.cn (H. Jin), ningxm@hust.edu.cn (X. Ning), itjia@cityu.edu.hk (W. Jia), haowu@hust.edu.cn (H. Wu), luguilin@gmail.com (G. Lu).

which return results in disorder that only meet query conditions with an exact boolean matching, it is more appropriate to formulate the above desired search results as a set of “degree of relevance” (e.g., in the interval $[0, 1]$). The “degree of relevance” is an overall score which should reflect both the multiple search properties and the subjective measurements thus able to support ranked queries. Moreover, as the number of ontologies available in the semantic web increases rapidly, it is reasonable to say that specialized ranking strategies will be indispensable for better satisfying user requirements to prevent providing users with many search results in disorder.

This paper proposes a novel approach which enables intelligent semantic web search for best satisfying users search intentions. The approach combines the user’s subjective weighting importance over multiple search properties together with fuzziness to represent search requirements. A special ranking mechanism based on the above weighed fuzzy query is also presented. The ranking method considers not only fuzzy predicates in the query, but also the user’s personalized interests or preferences. Therefore, the search results shall be properly ordered in terms of their “degree of relevance” with respect to the query request. In addition, the ranking method is general and unique rather than arbitrary. To verify its effectiveness, we evaluate the approach over real-world data from CiteSeer metadata (<http://citeseer.ist.psu.edu/oai.html>) which covers literatures in the field of computer science and computer technology.

The rest of this paper is organized as follows. Section 2 reviews related work. We detail our proposed weighted fuzzy approach in Section 3. Section 4 reports experimental results. Finally, we conclude the paper and point out future research directions in Section 5.

2. Related work

Efficient and intelligent semantic web search plays an important role in the vision of the semantic web [15,29,34,41]. Rocha et al. [34] present a semantic search [15] approach which combines traditional search engine techniques together with ontology based information retrieval. The approach differs from traditional search engines in that it seeks to find important related entities to a given set of keywords using a spreading activation algorithm [8] with the domain knowledge provided by experts. But the expressivity of its keywords based search model is very limited since it is unable to answer more complex structured queries. Additionally, their approach could not handle multiple vague search requirements and personalized preferences as our work does. In our previous work [29], we present a framework RSS which implements ranked semantic search on the semantic web. In this framework, search results can be greatly expanded with entities which are most semantically related to the query. Although the RSS query model is much more expressive than [34], it is unable to deal with more complex structured fuzzy queries. [41] implements a personalized content management system which is to improve the retrieval process by taking into account user preferences on the semantic context of ongoing user activities. In the context of personalized retrieval, the personal relevance measure is combined with query-dependent and user-neutral search result rank values to produce the final rank score for a document. Although the search process takes user preferences into account, the query model is still based on crisp logic and hence fails to represent vagueness.

In recent years, many impressive works have been carried out for tackling with imprecise and vague queries in the semantic web [38,44]. These approaches usually apply a so-called fuzzy description logic [40] which is an integration of Zadeh’s fuzzy logic [43] with classical DLs [2]. For example, Zhang et al. [44] proposes

a model which integrates information retrieval with formal fuzzy DL based query and reasoning to utilize textual and semantic information for searching in semantic portals. Similar to ours, the model also supports IR related formal queries like “find publications written by Tim Berners-Lee talking about semantic web”. However, their model could not deal with multiple fuzzy predicates since they only focused on textual information. It also lacks the important ranking capability as provided in our approach. Stoilos et al. [38] presents a fuzzy extension to OWL called Fuzzy OWL which can capture imprecise and vague knowledge. A prototype of fuzzy reasoning engine (FiRE) is also implemented to capture and reason about such fuzzy knowledge. FiRE does not support fuzzy datatype expressions [30] like “recent publications” or “highly cited publications” which are addressed by our work. Supporting flexible fuzzy queries has also been actively investigated in the databases area [6,23,27]. SQLf [6] allows imprecise querying of regular relational databases through extending the SQL language to satisfy user needs more closely. RankSQL [23] is a systematic and principled framework to support efficient evaluations of ranking (top-K) queries in relational database systems, by extending relational algebra and query optimization. In the system, ranking is taken as a first-class database construct which differs from SQLf. Meghini et al. [27] gives a fuzzy DL based multimedia information retrieval model which implements semantic based retrieval and offers a framework for the integration of the multimedia (including text and images) and multidimensional aspects (including form, content and structure). For personalized query in database system, Koutrika and Ioannidis [22] present a personalization framework for database queries based on information stored in structured user profiles that keep single, unconditional preferences. In this framework, a preference model assigns to each atomic query condition a personal degree of interest and then computes the degree of interest in complex query condition based on the degrees of interest in the constituent atomic ones.

To the best of our knowledge, the work that is most similar to ours is given in Ref. [36]. Siberski et al. [36] propose a comprehensive extension of SPARQL [37] which adds preference-based querying capabilities to SPARQL languages and hence directly supports the expression of preferences. They extend SPARQL with a new preferring solution sequence modifier. Therefore, users can specify which kind of answers they prefer by adding the preferring clauses to their queries like “FILTER (A or B) PREFERRING A”. But the search expression is still restricted to crisp logic. In addition, the effectiveness of this method is also unknown, since no evaluation results are given. Our work also relates to f-SWRL [31] which is a fuzzy extension of the expressive rule language SWRL [17] to include fuzzy assertions and fuzzy rules. In f-SWRL, atoms can include a “weight” (a truth-value between 0 and 1) that represents the “importance” of the atom in a rule. For example, the f-SWRL rule: $Tall(?x) * 0.7 \wedge Light(?x) * 0.8 \rightarrow Thin(?x)$, indicates that if one is Tall (with importance factor 0.7) and Light (with importance factor 0.8) then one is Thin. The semantics of weights are based on fuzzy aggregation functions, such as linear aggregation or weighted sum. Like [38], it also could not handle fuzzy datatypes, as we instead do.

3. Combining weights with fuzziness: our approach

This section details our approach for intelligent semantic web search. To make this paper self-explanatory, in Section 3.1 we first introduce fuzzy $\mathcal{ALC}(\mathbf{D})$ [39]. Then, Section 3.2 gives an overall architecture employing our approach, Section 3.3 describes the underlying fuzzy knowledge base, Section 3.4 presents user queries formulation and finally Section 3.5 depicts our ranking mechanism and retrieval.

3.1. Fuzzy $\mathcal{ALC}(\mathbf{D})$

We briefly review the fuzzy $\mathcal{ALC}(\mathbf{D})$ [39] which is a fuzzy version of the basic yet expressive DL \mathcal{ALC} [35] with concrete domains [26]. We assume that our reader is already familiar with the basics of DLs [2,18] and fuzzy theory [43]. Regarding to the integration of classical DLs with fuzzy theory, Straccia [40] presents a fuzzy description logic fuzzy \mathcal{ALC} which is a combination of fuzzy logic with \mathcal{ALC} . However, fuzzy \mathcal{ALC} does not support fuzzy concrete domains which is fully employed in our model.

3.1.1. Definition of syntax

In fuzzy logics, the degree of membership $S(u)$ is regarded as the *truth-value* (between 0 and 1) of the statement “ u is S ”. Accordingly, for fuzzy DLs, a concept C is interpreted as a fuzzy set rather than as a classical set, and the statement “ a is C ” (i.e., $a : C$) has a *truth-value* given by the membership degree $C(a)$. Switching to fuzzy $\mathcal{ALC}(\mathbf{D})$, concrete domains are also interpreted as fuzzy sets. A *fuzzy concrete domain* \mathbf{D} is a pair $\mathbf{D} = \langle \Delta_{\mathbf{D}}, \Phi_{\mathbf{D}} \rangle$, where $\Delta_{\mathbf{D}}$ is an interpretation domain and $\Phi_{\mathbf{D}}$ is the set of *fuzzy domain predicates* d with a predefined arity n and an interpretation $d^{\mathbf{D}} : \Delta_{\mathbf{D}}^n \rightarrow [0, 1]$, which is a n -ary fuzzy relation over $\Delta_{\mathbf{D}}$. For example, in the scientific literature domain, the fuzzy domain predicate **Recent** may denote the degree of recentness of a specific publication with the following membership function:

$$\text{Recent}(x) = \begin{cases} 0 & \text{if } x < a \\ \frac{x-a}{b-a} & \text{if } a \leq x < b \\ 1 & \text{if } x \geq b \end{cases} \quad (1)$$

where x is the publishing year, a and b might be set as a past specified year (e.g., 1946, 1980 or 2000, etc., depending on application areas like *History* or *Computer Science*) and the present year (i.e., 2007), respectively. Assume that a paper was published in 2001 (i.e., $x = 2001$), and we set $a = 1980$, $b = 2007$. Hence, we can say that this paper is a recent publication with a degree $\text{Recent}(x = 2001) = 0.78$.

Let C , R_a , R_c , I_a and I_c be non-empty finite and pair-wise disjoint sets of *concepts names* (denoted A), *abstract roles names* (denoted R), *concrete roles names* (denoted T), *abstract individual names* (denoted a) and *concrete individual names* (denoted c). R_a contains a non-empty subset F_a of *abstract feature names* (denoted r), while R_c contains a non-empty subset F_c of *concrete feature names* (denoted t). Features are functional roles. The set of fuzzy $\mathcal{ALC}(\mathbf{D})$ concepts is defined by the following syntactic rules (for simplicity, d is restricted to a unary fuzzy predicate):

$$\begin{aligned} C &\rightarrow \perp \mid \top \mid A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \neg C \mid \forall R.C \mid \exists R.C \mid \forall T.D \mid \exists T.D \\ D &\rightarrow d \mid \neg d \end{aligned}$$

A fuzzy $\mathcal{ALC}(\mathbf{D})$ knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ consists of a TBox \mathcal{T} and an ABox \mathcal{A} . A TBox \mathcal{T} axioms are of the form $A \sqsubseteq C$ or $A = C$, where A and C are all concepts. For instance, we may define the concept of a recent publication as

$$\text{RecentPublication} = \text{Publication} \sqcap \exists \text{year.Recent}$$

where **year** $\in R_c$ is a concrete role and **Recent** is a unary fuzzy predicate. In an ABox \mathcal{A} , *fuzzy assertions* are of the form $\langle \alpha, n \rangle$, where α is a classical concept or role assertion in the form of $a : C$ or $(a, b) : R$ and $n \in [0, 1]$ represents the truth degree of α to be greater than n . For example, the fuzzy assertion $\langle p : \text{RecentPublication} \geq 0.7 \rangle$ indicates that paper p is a recent publication with a degree greater than 0.7.

3.1.2. Model-theoretic semantics

A *fuzzy interpretation* \mathcal{I} with respect to a fuzzy concrete domain $\mathbf{D} = \langle \Delta_{\mathbf{D}}, \Phi_{\mathbf{D}} \rangle$ is a tuple $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ which consists of a non-empty

(abstract) domain $\Delta^{\mathcal{I}}$, disjoint from $\Delta_{\mathbf{D}}$ and a *fuzzy interpretation function* $\cdot^{\mathcal{I}}$ that assigns (1) to each abstract concept $C \in C$ a function $C^{\mathcal{I}} : \Delta^{\mathcal{I}} \rightarrow [0, 1]$; (2) to each abstract role $R \in R_a$ a function $R^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \rightarrow [0, 1]$; (3) to each abstract feature $r \in F_a$ a partial function $r^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \rightarrow [0, 1]$ such that for all $u \in \Delta^{\mathcal{I}}$ there is an unique $w \in \Delta^{\mathcal{I}}$ on which $r^{\mathcal{I}}(u, w)$ is defined; (4) to each abstract individual $a \in I_a$ an element in $\Delta^{\mathcal{I}}$; (5) to each concrete individual $c \in I_c$ an element in $\Delta_{\mathbf{D}}$; (6) to each concrete role $T \in R_c$ a function $T^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta_{\mathbf{D}} \rightarrow [0, 1]$; (7) to each concrete feature $t \in F_c$ a partial function $t^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta_{\mathbf{D}} \rightarrow [0, 1]$ such that for all $u \in \Delta^{\mathcal{I}}$ there is an unique $o \in \Delta_{\mathbf{D}}$ on which $t^{\mathcal{I}}(u, o)$ is defined; (8) to each unary concrete predicate d the fuzzy relation $d^{\mathbf{D}} : \Delta_{\mathbf{D}} \rightarrow [0, 1]$ and to $\neg d$ the negation of $d^{\mathbf{D}}$. In addition, the *fuzzy interpretation* \mathcal{I} should satisfy the following equations (where $u \in \Delta^{\mathcal{I}}$):

$$\begin{aligned} \perp^{\mathcal{I}}(u) &= 0 \\ \top^{\mathcal{I}}(u) &= 1 \\ (C_1 \sqcap C_2)^{\mathcal{I}}(u) &= C_1^{\mathcal{I}}(u) \wedge C_2^{\mathcal{I}}(u) \\ (C_1 \sqcup C_2)^{\mathcal{I}}(u) &= C_1^{\mathcal{I}}(u) \vee C_2^{\mathcal{I}}(u) \\ (\neg C)^{\mathcal{I}}(u) &= 1 - C^{\mathcal{I}}(u) \\ (\forall R.C)^{\mathcal{I}}(u) &= \inf_{w \in \Delta^{\mathcal{I}}} R^{\mathcal{I}}(u, w) \rightarrow C^{\mathcal{I}}(w) \\ (\exists R.C)^{\mathcal{I}}(u) &= \sup_{w \in \Delta^{\mathcal{I}}} R^{\mathcal{I}}(u, w) \wedge C^{\mathcal{I}}(w) \\ (\forall T.D)^{\mathcal{I}}(u) &= \inf_{o \in \Delta_{\mathbf{D}}} T^{\mathcal{I}}(u, o) \rightarrow D^{\mathcal{I}}(o) \\ (\exists T.D)^{\mathcal{I}}(u) &= \sup_{o \in \Delta_{\mathbf{D}}} T^{\mathcal{I}}(u, o) \wedge D^{\mathcal{I}}(o) \end{aligned}$$

A fuzzy interpretation \mathcal{I} *satisfies* (is a model of) a fuzzy assertion γ (written $\mathcal{I} \models \gamma$) defined by: $\mathcal{I} \models A \sqsubseteq C$ iff for all $u \in \Delta^{\mathcal{I}}$, $A^{\mathcal{I}}(u) \leq C^{\mathcal{I}}(u)$; $\mathcal{I} \models A = C$ iff for all $u \in \Delta^{\mathcal{I}}$, $A^{\mathcal{I}}(u) = C^{\mathcal{I}}(u)$; $\mathcal{I} \models \langle a, n \rangle$ iff $a^{\mathcal{I}} \geq n$; $\mathcal{I} \models a \approx b$ iff $a^{\mathcal{I}} = b^{\mathcal{I}}$; $\mathcal{I} \models a \neq b$ iff $a^{\mathcal{I}} \neq b^{\mathcal{I}}$. An interpretation \mathcal{I} is a *model* of a fuzzy $\mathcal{ALC}(\mathbf{D})$ KB \mathcal{K} iff \mathcal{I} satisfies every element of \mathcal{K} . A fuzzy KB \mathcal{K} *entails* a fuzzy assertion γ (written $\mathcal{K} \models \gamma$) iff every model of \mathcal{K} satisfies γ .

Particularly we note that, unlike previous fuzzy DLs [40] which use the classic fuzzy logic interpretation on constructors, e.g., conjunction (\wedge) as min and disjunction (\vee) as max, our approach interprets these constructors differently. This is due to the observation that for different applications, the standard min based function may lead to insufficiently discriminative ranking results since it only draws upon the comparison of the least satisfied properties. The above observation is exploited in our ranking algorithm.

3.2. Overview of approach

The main idea in our approach is that, the user's search request which may contain fuzziness and complex personalized preferences is first formulated to a formal query, then this query is submitted to a search engine backed by a fuzzy $\mathcal{ALC}(\mathbf{D})$ KB, and then initial search results generated from the search engine are properly ordered via a special ranking mechanism.

Fig. 1 illustrates the overview architecture employing our approach. The architecture consists of the following functionality components: (1) the *Query UI* that provides user query interface; (2) the *Query Formulation* module that parses the user query and composes the formal query; (3) the *Fuzzy Search/Inference Engine* that answers fuzzy query; (4) the *Fuzzy KB* (with rules) that reposit fuzzy $\mathcal{ALC}(\mathbf{D})$ knowledge; (5) the *Full-text Engine* that provides full-text indexing and searching; (6) and the *Ranking Module* that produces the final ranked search results. In particular, the *Full-text Engine* is an external textual search engine which helps to handle IR related fuzzy queries like “talking about description logic” or similar document retrievals like “publications similar to a specified paper p ”. The similarity function and membership functions are used to interpret fuzzy concrete domain predicates (e.g., *Recent*, *Highly*, *SimilarTo*) which are to be detailed in Section 3.3.

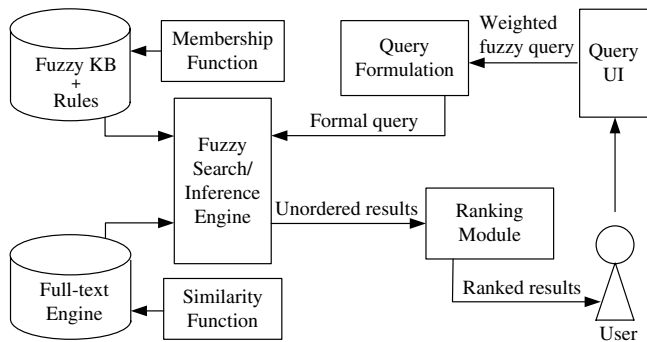


Fig. 1. Overview architecture.

The basic steps of processing a user query is here outlined:

- (1) The user submits a query which may contain fuzziness and his own subjective weighting measurements via *Query UI*.
- (2) The *Query Formulation* module formats the query to a structured formal query.
- (3) The formal query is handled by *Fuzzy Search/Inference Engine*, and initial unordered results are obtained.
- (4) Initial results are further processed by the *Ranking Module*, and then final ranked results are provided to the user.

3.3. Underlying knowledge base

To explain the applicability of our method, we select the scientific literature domain as the knowledge base. But we note that it is also appropriate to extend it to other application domains, such as multimedia retrieval [12,27], semantic web services matchmaking

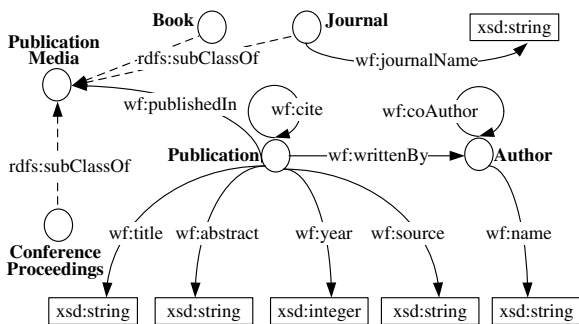
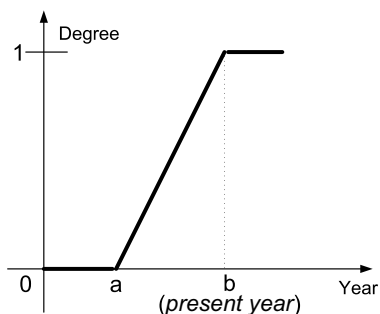
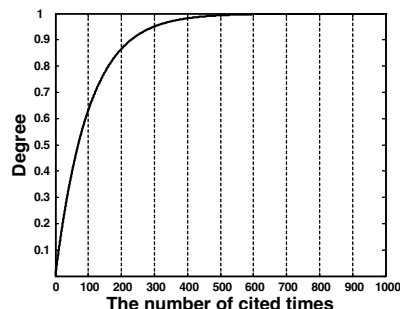


Fig. 2. A simplified OWL representation of scientific literature domain.



(a) Membership function of **Recent**



(b) Membership function of **Highly**

Fig. 3. Examples of membership functions for fuzzy concrete domain predicates.

and composition [24], and digital libraries. The selected KB mainly concerns concepts, roles and individuals about scientific literatures. Fig. 2 shows a simplified ontology representation of the scientific literature domain (*wf* is the local namespace). Some elements are omitted from the figure for the ease of presentation.

As depicted in Fig. 2, the major concepts in the application domain are *Author*, *Publication* and *PublicationMedia*. The concept *PublicationMedia* has several subClasses, such as *Journal*, *Book*, *ConferenceProceedings* and *TechnicalReport*. The roles include object properties (e.g., *writtenBy*, *cite*, *publishedIn*, *coAuthor*) and datatype properties (e.g., *title*, *abstract*, *year*, *journalName*). Particularly, the role *source* is used to record the full-text source of a publication. Textual information is handled by the extern *Full-text Engine* since current RDF/OWL storage and retrieval systems (e.g., Sesame, Jena [20] and KAON2) are inefficient on full-text processing.

Next, we focus on the representation of fuzzy predicates, i.e., *Recent*, *Highly*, *IsAbout* and *SimilarTo*. For example, in current digital libraries (e.g., IEEE Xplore), if one wants to find recently published papers which full-text is about semantic web, he may first specify the year (e.g., 2007) or a year range (e.g., from 2005 to 2007) and then enter the keywords “semantic web”. Then those papers published in 2004 or before will be discarded even if there some of them might be very relevant to “semantic web”. Therefore, it is better to give a degree on whether a paper belongs to a recent publication rather than an exact boolean matching. The same holds for the predicate *Highly* which indicates that a publication is cited many times by other ones since it is inappropriate to simply represent those highly publications as (*SHOIN*(**D**) syntax) $\text{Publication} \sqcap \geq 100\text{citedBy}$. The selection of membership functions for fuzzy predicates is application and context dependent [45]. Consider the membership function of *Recent* defined in Section 3.1. Given a corpus of publications where all publishing years are almost between 1990 and 2005, we may set $a = 1990$ and $b = 2005$ instead. The membership function of *Highly* is different from that of *Recent*. As the number of cited times can be used to indicate the paper’s impact to some degree, we might apply the following membership function:

$$\text{Highly}(n) = 1 - \exp\left(-\frac{n}{100}\right) \quad (2)$$

where n is the cited times of a specific publication. Fig. 3 gives geometric forms for the two fuzzy domain predicates. From Fig. 3b, we can see that the value of degree increases rapidly as the number of cited times n increases from 0 to 100. After that, the increase becomes not very apparent, especially when $n > 400$. This phenomenon accords with the fact that, if a paper is cited too many times, more citations might have little influence on its impact, such as the pioneering paper *The semantic web* by Tim Berners-Lee et al. [4]. With the introduction of the fuzzy predicate *Highly*, we can define the fuzzy concept of a highly cited publication as follows:

HighlyCitedPublication = Publication \square \exists timesCited.Highly

Now, we describe the other two fuzzy predicates *IsAbout* and *SimilarTo*, to handle IR related queries. Given a IR query string s and a publication d , the predicate *IsAbout* indicates the fuzzy degree of relevance between s and d . Techniques like the extended boolean approach and the probabilistic approach, can be used to determine the value. In this paper, we will apply the popular cosine similarity measure of vector space model (VSM) [5]. Let \vec{s} be the query vector of s and \vec{t} be the document vector of d , respectively, where $d.t$ denotes the textual content of d . The semantics of *IsAbout* is interpreted via the fuzzy $\mathcal{ALC}(\mathbf{D})$ interpretation \mathcal{I} :

$$\text{IsAbout}^{\mathcal{I}}(d, s) = \frac{t \cdot s}{\|\vec{t}\| \cdot \|\vec{s}\|} \rightarrow [0, 1] \quad (3)$$

The predicate *SimilarTo*, which determines the degree of relevance between two publications, is defined analogously.

Our knowledge base can be easily extended with DL-safe rules [28], for example:

$$\begin{aligned} \mathcal{O}(x), \mathcal{O}(y), \mathcal{O}(z), \text{Publication}(z), \text{writtenBy}(z, x), \\ \text{writtenBy}(z, y) \rightarrow \text{coAuthor}(x, y) \\ \mathcal{O}(x), \mathcal{O}(y), \mathcal{O}(z), \text{Publication}(z), \text{citedBy}(x, z), \\ \text{citedBy}(y, z) \rightarrow \text{coCitation}(x, y) \end{aligned}$$

where \mathcal{O} is used to bound each variable only to individuals explicitly introduced in the ABox. But the work of incorporating fuzzy domain predicates into rules, is out of the scope of this paper and is considered as our future work.

3.4. User queries formulation

After describing our fuzzy knowledge base, we are now in the position to formally formulate user queries. Recall the motivating query by the Ph.D. candidate who aims to find *recent* publications *highly cited* *written by* Zadeh, but *prefer* more weight on recentness over citation (see Section 1). The query can be splitted into two parts: “*recent* publications *highly cited* *written by* Zadeh” and “*prefer* more weight on recentness over citation”. Accordingly, we proceed the formulation task with two major steps. We first tackle the former part. This procedure requires: (1) the representation of fuzzy predicates (i.e., *recent* and *highly*) and crisp (boolean) predicates (i.e., *writtenBy*); (2) a so-called *aggregation function* which correctly interprets the conjunctive constructor, such as **min** in classic Zadeh fuzzy logic, **product** in product logic, **averaged sum** [45]. Then, we process the latter part which involves the user’s subject weights over two fuzzy predicates. The weights should be smoothly incorporated into the *aggregation function* so as to give an overall score that reflect both the multiple predicates and the weights over them. The computation of the overall scores, is to be addressed in Section 3.5.

Definition 1 (Conjunctive Query). Let KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ with TBox \mathcal{T} and the ABox \mathcal{A} , be a fuzzy $\mathcal{ALC}(\mathbf{D})$ knowledge base, and let N_p be a set of predicates which includes all concepts and all abstract roles and all concrete roles in \mathcal{K} , i.e., $N_p = C \cup R_a \cup R_c$. Let $\mathbf{x} = \{x_1, \dots, x_n\}$ and $\mathbf{y} = \{y_1, \dots, y_m\}$ be sets of *distinguished (free)* and *non-distinguished (existentially quantified)* variables, respectively. An atom has the form $P(\mathbf{s}) = P(s_1, \dots, s_l)$, where $P \in N_p$ and s_i are either variables or individuals from \mathcal{K} . Let p_i be a k_i -ary fuzzy predicate which assigns each k_i -ary tuple \mathbf{e}_i to a value $p_i(\mathbf{e}_i) \in [0, 1]$, where \mathbf{e}_i is a tuple of variables in \mathbf{x} and \mathbf{y} or constants in \mathcal{K} . Let $f = f(p_1(\mathbf{e}_1), \dots, p_k(\mathbf{e}_k))$ be an *aggregation function* $f : [0, 1]^k \rightarrow [0, 1]$, where f depends on the type of interpretation on the conjunctive constructor (\wedge) such as *min*, *product*, *averaged sum*, etc. The *aggregation function* f is used to combine the scores of k fuzzy predicates $p_i(\mathbf{e}_i)$ into an overall score (so far, without consideration

on weights). A conjunctive query $Q(\mathbf{x}, f)$ over \mathcal{K} has the following expression:

$$Q(\mathbf{x}, f) = \exists \mathbf{y} : \bigwedge P_i(s_i) \text{ AND } f$$

where AND is only a symbol which denotes that f is associated with the query $Q(\mathbf{x}, f)$.

The choice of aggregation function f is application specific [45]. For example, in the application on scheduling large scale on-demand data broadcast [1], the authors make use of the product function with $f(x_1, x_2) = x_1 \cdot x_2$ as they wish to broadcast next the page with the top score. In addition, since a *disjunctive query* is a set of conjunctive queries where heads (i.e., $Q(\mathbf{x}, f)$) are the same, we concentrate on conjunctive queries in this paper.

Regarding to query answering, an answer of a conjunctive query $Q(\mathbf{x}, f)$ is the set of tuples $\langle \theta, v \rangle$, where θ are substitutions of *distinguished* variables \mathbf{x} with individuals in \mathcal{K} , and $v = f(p_1(\mathbf{e}'_1), \dots, p_k(\mathbf{e}'_k)) \in (0, 1]$ in which \mathbf{e}'_i are substitutions of \mathbf{e}_i .

Example. With the definition of conjunctive queries, we now can formally represent the motivating query (only the former part) as follows:

$$\begin{aligned} Q(\mathbf{x}, f) = \text{Publication}(x_1) \wedge \text{year}(x_1, y_1) \wedge \text{timesCited}(x_1, y_2) \\ \wedge \text{writtenBy}(x_1, y_3) \text{ AND } f \end{aligned}$$

where $\mathbf{x} = \{x_1\}$, $y_3 = \{\text{“Zadeh”}\}$, $\mathbf{y} = \{y_1, y_2, y_3\}$ and $f = f(\text{Recent}(y_1), \text{Highly}(y_2))$. To further explain the definition, we give a more concrete example. Assume that, a publication of Zadeh was published in 2001 and was cited 200 times until now. Hence, x_1 represents the mentioned publication (an individual in \mathcal{K}), $y_1 = 2001$ and $y_2 = 200$. We apply Eq. (1) (where $a = 1980$ and $b = 2007$) and Eq. (2) to obtain $\text{Recent}(y_1 = 2001) = \frac{2001-1980}{2007-1980} = 0.78$ and $\text{Highly}(y_2 = 200) = 1 - \exp(-\frac{200}{100}) = 0.86$, respectively. Therefore, we get $f = \min(0.78, 0.86) = 0.78$ with the interpretation **min** on conjunctive constructor (\wedge), or $f = \frac{0.78+0.86}{2} = 0.82$ with **averaged sum** instead.

Next, we define the notion of extended conjunctive queries which are equipped with weights.

Definition 2 (Extended Conjunctive Query). Assume that $\lambda_1, \dots, \lambda_k$ satisfy the following equation:

$$\sum_{i=1}^k \lambda_i = 1, \quad \text{where } \lambda_i \in [0, 1] \quad (4)$$

We define $\Lambda = (\lambda_1, \dots, \lambda_k)$ as a weighting. Given a conjunctive query $Q(\mathbf{x}, f)$ without weights where $f = f(p_1(\mathbf{e}_1), \dots, p_k(\mathbf{e}_k))$, a weighting Λ is assigned to the k fuzzy predicates such that λ_i is the weight of fuzzy predicate p_i . A *weighting function* f_Λ is defined to give an overall score (between 0 and 1) which reflects both fuzzy predicates and weights in the query. We give the explicit definition of the weighting function f_Λ in Section 3.5. Now, we can define the concept of an extended conjunctive query as follows:

$$Q(\mathbf{x}, f_\Lambda) = \exists \mathbf{y} : \bigwedge P_i(s_i) \text{ AND } f_\Lambda$$

Here, we use a qualitative weighting method (i.e., Λ) to express the user’s subjective measures on multiple fuzzy predicates. This may require the query interface (i.e., *Query UI* in Fig. 1) to provide a form where the end user can assign numerical weights to different properties, and hence impose some cognitive overhead on users since they sometimes may have difficulties in expressing their information needs in a precisely quantitative way. There have been attempts to seek more user-friendly query languages, though still very complex. For example, [16] proposes an ordinal fuzzy linguistic approach for information retrieval. In their approach, numerical weights are replaced by an ordered lin-

guistic term set like $S = \{none, verylow, low, medium, high, veryhigh, perfect\}$. Then, these qualitative terms are transformed into quantitative numbers via a so-called evaluation subsystem [16]. There are also efforts for automatic weights assignment based on collected user profiles. A personalization system [22] is presented for database queries based on user profiles which are inserted explicitly by the user or collected implicitly by monitoring user interaction with the system. But to reduce the complexity of our problem, we still remain the qualitative weighting method in this paper.

The answer of an extended conjunctive query is analogous to that of an conjunctive query, except that the *aggregation function* f is accordingly replaced by the *weighting function* f_Λ .

Example. With the above definitions, we can give the formulation for the motivating query as a whole. We assume that, the Ph.D. candidate would like to put twice weight as much on recentness as on citation. Thus, we get the weighting $\Lambda = (\lambda_1 = \frac{2}{3}, \lambda_2 = \frac{1}{3})$ according to Eq. (4), where λ_1 and λ_2 are the weights assigned to fuzzy predicates *Recent*(y_1) and *Highly*(y_2), respectively. The extended conjunctive query has the following expression of the form:

$$Q(\mathbf{x}, f_\Lambda) = \text{Publication}(x_1) \wedge \text{year}(x_1, y_1) \wedge \text{timesCited}(x_1, y_2) \\ \wedge \text{writtenBy}(x_1, y_3) \text{ AND } f_\Lambda$$

where f_Λ is the weighting function which is to be detailed in the next section.

3.5. Ranking algorithm

Recall that an answer of an extended conjunctive query $Q(\mathbf{x}, f_\Lambda)$ is the set of tuples $\langle \theta, v \rangle$, where *distinguished* variables \mathbf{x} are substituted with individuals θ in \mathcal{K} and $v = f_\Lambda \in (0, 1]$. For a real-world fuzzy $\mathcal{ALC}(\mathbf{D})$ KB \mathcal{K} , the ABox \mathcal{A} may be huge and hence there may be many possible substitutions (i.e., query results) for \mathbf{x} . Therefore, a specialized ranking mechanism is necessary such that query results are non-increasingly ordered according to their “*degree of relevance*” to the query. This ranking mechanism is implemented with the help of the *weighting function* f_Λ . For the rest of this section, we will detail it. Moreover, we shall show that the definition of f_Λ is unique rather than arbitrary. The proposed *weighting function* is also general and is not restricted to just *min*, *product* or *averaged sum*.

Intuitively, since $f = f(p_1(\mathbf{e}_1), \dots, p_k(\mathbf{e}_k))$ and $\Lambda = (\lambda_1, \dots, \lambda_k)$ where λ_i is the weight of p_i , we might simply assume a *weighting function* $f_\Lambda = \lambda_1 \cdot p_1(\mathbf{e}_1) + \dots + \lambda_k \cdot p_k(\mathbf{e}_k)$ or $f_\Lambda = \min(\lambda_1 \cdot p_1(\mathbf{e}_1), \dots, \lambda_k \cdot p_k(\mathbf{e}_k))$. Unfortunately, these naive assumptions have serious flaws. For instance, they might not properly handle the problem that, if all weights are equal, then the extended conjunctive query should return the same query results as the conjunctive query without a weighting. Regarding to the former assumption, we assume that the Ph.D. candidate would like to find *recent* publications *highly* cited *written* by Zadeh, but indifferent to *recentness* vs. *citation*, that is, he puts **the same weight** as much on recentness as on citation. So we have $f = f(\text{Recent}(y_1), \text{Highly}(y_2))$ and $\Lambda = (\frac{1}{2}, \frac{1}{2})$. According to the above assumption, we could get $f_\Lambda = \frac{1}{2} \cdot \text{Recent}(y_1) + \frac{1}{2} \cdot \text{Highly}(y_2) = \frac{\text{Recent}(y_1) + \text{Highly}(y_2)}{2}$. If **averaged sum** is used to interpret the conjunctive constructor (\wedge), then $f = \frac{\text{Recent}(y_1) + \text{Highly}(y_2)}{2} \equiv f_\Lambda$. However, if we use **min** instead, then $f = \min(\text{Recent}(y_1), \text{Highly}(y_2)) \not\equiv f_\Lambda$. Therefore, the former naive *weighting function* is only suitable for **averaged sum**. Analogously, we can show that the latter assumption is even not suitable for either **averaged sum** or **min**.

Next, we give the formal definition. A desirable *weighting function* f_Λ should satisfy the following properties:

- (1) If $\Lambda = (\frac{1}{k}, \dots, \frac{1}{k})$, then $f_\Lambda(p_1(\mathbf{e}_1), \dots, p_k(\mathbf{e}_k)) = f(p_1(\mathbf{e}_1), \dots, p_k(\mathbf{e}_k))$. This property indicates that, if all of the weights in Λ are equal, then the weighting function f_Λ should be equal to the aggregation function f which does not consider weights.
- (2) If $\Lambda = (\lambda_1, \dots, \lambda_{k-1}, 0)$, then $f_\Lambda(p_1(\mathbf{e}_1), \dots, p_k(\mathbf{e}_k)) = f_{\Lambda'}(p_1(\mathbf{e}_1), \dots, p_{k-1}(\mathbf{e}_{k-1}))$ where Λ' is a weighting $\Lambda' = (\lambda_1, \dots, \lambda_{k-1})$. This property is used to handle zero weights. That is, if a particular fuzzy predicate has zero weight, then that fuzzy predicate can be dropped without affecting the value of the result.
- (3) For $\Lambda = (\lambda_1, \dots, \lambda_k)$, $f_\Lambda(p_1(\mathbf{e}_1), \dots, p_k(\mathbf{e}_k))$ is a continuous function of $\lambda_1, \dots, \lambda_k$. This property is natural since a small change in the weights should only lead to at most a small change in the value of the result.
- (4) The final property is about *linearity*. Given two weightings $\Lambda = (\lambda_1, \dots, \lambda_k)$ and $\Lambda' = (\lambda'_1, \dots, \lambda'_k)$, if they agree on the order of weights, that is, there do not exist i, j with $\lambda_i < \lambda_j$ and $\lambda'_j < \lambda'_i$ both holding, then,

$$f_{\alpha\Lambda + (1-\alpha)\Lambda'}(P) = \alpha \cdot f_\Lambda(P) + (1-\alpha) \cdot f_{\Lambda'}(P), \text{ where } \alpha \in [0, 1] \text{ and} \\ P = (p_1(\mathbf{e}_1), \dots, p_k(\mathbf{e}_k))$$

This property is a little complex, however, it is also natural. It indicates that, the weighting (i.e., $\alpha \cdot \Lambda + (1-\alpha) \cdot \Lambda'$) that is the midpoint of the above two weightings (i.e., Λ and Λ') should produce a value that is the midpoint of the two values produced by the given weightings.

The above properties are necessary for the *weighting function* f_Λ , and any other *weighting function* w.r.t. a given *aggregation function* that does not satisfy them has serious flaws. Given these properties, the *weighting function* f_Λ is to be uniquely determined. For a weighting $\Lambda = (\lambda_1, \dots, \lambda_k)$, without loss of generality, we assume $\lambda_1 \geq \dots \geq \lambda_k$, then

$$f_\Lambda(p_1(\mathbf{e}_1), \dots, p_k(\mathbf{e}_k)) = (\lambda_1 - \lambda_2) \cdot f(p_1(\mathbf{e}_1)) \\ + 2(\lambda_2 - \lambda_3) \cdot f(p_1(\mathbf{e}_1), p_2(\mathbf{e}_2)) \\ + 3(\lambda_3 - \lambda_4) \cdot f(p_1(\mathbf{e}_1), p_2(\mathbf{e}_2), p_3(\mathbf{e}_3)) \\ + \dots + k\lambda_k \cdot f(p_1(\mathbf{e}_1), \dots, p_k(\mathbf{e}_k)) \\ = k\lambda_k \cdot f(p_1(\mathbf{e}_1), \dots, p_k(\mathbf{e}_k)) \\ + \sum_{i=1}^{k-1} i(\lambda_i - \lambda_{i+1}) \cdot f(p_1(\mathbf{e}_1), \dots, p_i(\mathbf{e}_i)) \quad (5)$$

For the detailed proof of this formula, the reader may refer to [11] which implements fuzzy queries for a multimedia database system.

Let us give examples to explain the proposed *weighting function* f_Λ .

Example. A user issues the following query: finding *recent* publications *highly* cited talking *about* fuzzy logic written by Zadeh, and putting weights 0.5, 0.3 and 0.2 to fuzzy predicates *Recent*, *Highly* and *IsAbout*, respectively. The query can be represented as:

$$Q(\mathbf{x}, f_\Lambda) = \text{Publication}(x_1) \wedge \text{year}(x_1, y_1) \wedge \text{timesCited}(x_1, y_2) \wedge \text{isLiteral}(y_3) \\ \wedge \text{writtenBy}(x_1, y_4) \text{ AND } f_\Lambda$$

where $\Lambda = (\lambda_1 = 0.5, \lambda_2 = 0.3, \lambda_3 = 0.2)$, the query keywords $y_3 =$ fuzzy logic and the author $y_4 = \{\text{Zadeh}\}$. For simplicity, we use the following notations: $z_1 = \text{Recent}(y_1)$, $z_2 = \text{Highly}(y_2)$ and $z_3 = \text{IsAbout}(x_1, y_3)$. Hence, we get the aggregation function $f = f(z_1, z_2, z_3)$. We assume that there exists a specific publication x_1 satisfying the above query and having $\text{Recent}(y_1) = 0.78$, $\text{Highly}(y_2) = 0.86$ and $\text{IsAbout}(x_1, y_3) = 0.70$. According to Eq. (5), we have:

$$f_\Lambda = (\lambda_1 - \lambda_2) \cdot f(z_1) + 2(\lambda_2 - \lambda_3) \cdot f(z_1, z_2) + 3\lambda_3 \cdot f(z_1, z_2, z_3)$$

The choice of aggregation function f is application specific [45]. For example, if **min** is required to interpret the aggregation function f , then by expanding the above equation we get:

$$\begin{aligned} f_{\Lambda} &= (\lambda_1 - \lambda_2) \cdot z_1 + 2(\lambda_2 - \lambda_3) \cdot \min(z_1, z_2) + 3\lambda_3 \cdot \min(z_1, z_2, z_3) \\ &= (0.5 - 0.3) \cdot 0.78 + 2(0.3 - 0.2) \cdot \min(0.78, 0.86) \\ &\quad + 3 \cdot 0.2 \cdot \min(0.78, 0.86, 0.70) \\ &= 0.732 \end{aligned}$$

Instead, if **averaged sum** is needed to interpret f , we will have:

$$f_{\Lambda} = (\lambda_1 - \lambda_2) \cdot z_1 + 2(\lambda_2 - \lambda_3) \cdot \frac{z_1 + z_2}{2} + 3\lambda_3 \cdot \frac{z_1 + z_2 + z_3}{3} = 0.788$$

The result for **product** can be gotten analogously, except that $f(z_1, \dots, z_k) = z_1 \cdots z_k$, $1 \leq k \leq 3$.

Finally, we briefly discuss query results since parts of this work have been given in Section 3.4. As mentioned previously, for a huge fuzzy $\mathcal{A}^{\mathcal{L}}\mathcal{C}(\mathbf{D})$ knowledge base, there may be too many possible substitutions (i.e., query results) with a non-zero value (i.e., *degree of relevance*) calculated by the *weighting function* f_{Λ} . These results can be non-increasingly ordered according to their values thus able to support *ranked queries*. In addition, some techniques could be used to improve the quality of the final ranking, e.g., setting a threshold value such that those values under the threshold are to be discarded, or imposing a limit on the total output length (namely, the so-called top- K queries). But we do not discuss these techniques further here.

4. Experiments

The major goal of the experiments is to show the effectiveness of our proposed approach. We first describe the datasets and the environmental setup for the experiments in Section 4.1. To best understand our approach and clarify some of the rather formal definitions introduced in our approach, we then give several experimental examples in Section 4.2. Finally, we discuss the experimental results in Section 4.3.

4.1. Datasets and experimental setup

Our data primarily comes from CiteSeer metadata (<http://cite-seer.ist.psu.edu/oai.html>), including 12 CiteSeer BibTex records (CiteSeerBibTex N .zip, $1 \leq N \leq 12$) and the metadata archive (oai_cite-seer.tar.gz) which is compliant with the Dublin Core standard with additional corresponding metadata fields (e.g., abstract, citation relationships, author affiliations and author addresses). The metadata covers literatures in the field of computer science and computer technology with about 800,000 publications and totals approximately 2 GB. To verify the effectiveness of our approach, we mainly select information systems related data. In particular, we use (wildcard) keywords “knowledge”, “data”, “database”, “intelligen”, “information s”, “AI” or “information retrieval” to match the title or the booktitle (i.e., inproceedings) or the journal (i.e., article) field of BibTex entries. Most publications within the information systems area are filtered, such as database, artificial intelligence, data mining, information storage and retrieval, information systems applications. During the above filtering procedure, some BibTex entries are discarded as errors, e.g., title or author or url missing. Then, we combine the filtered BibTex entries with the metadata archive and generate detail information on publications, including unique id, title, author, abstract, citation relationship, publisher and source (i.e., web URL of the original publication).

We build an OWL ontology schema (i.e., TBox \mathcal{T}) for representing the scientific literature domain as described in Fig. 2 using

Protégé 3.2.1 [32]. Then, the detail information on publications is encoded with Jena 2.5.2 [20] to form the ABox \mathcal{A} . To handle IR related fuzzy predicates (i.e., *IsAbout*, *SimilarTo*), we use Apache Lucene 2.1.0 [25] as the *Full-text Engine* because current RDF/OWL storage and retrieval systems (e.g., Sesame, Jena and KAON2) are inefficient on full-text processing. The class *org.apache.lucene.search.Hits* in Lucene API is employed to compute the similarity between the textual content of a publication d and an IR query string s , namely the fuzzy predicate *IsAbout* $^f(d, s)$. The full-texts of publications could have been downloaded from the web via the source field. But in the experiments, we combine the abstract and title fields as the textual content of a publication since the two important fields can usually reflect its textual information in a certain degree. In the knowledge base, there are totally 459,262 RDF triples, 18,183 *Publication* individuals, 15,286 *author* individuals, 447 distinct *Publication Media* individuals (including 86 *Journal* individuals and 361 *ConferenceProceedings* individuals), and 142.5 MB textual contents indexed by Lucene. Regarding to the membership function *Recent*(x), since the CiteSeer metadata archive is not up-to-date and most publications in the archive were published between 1990 and 2003, we set $a = 1990$ and $b = 2003$ in Eq. (1) for the following experiments. In addition, the membership function *Highly*(n) is set as $\text{Highly}(n) = 1 - \exp(-\frac{n}{30})$ where n is the cited times of a publication.

4.2. Experimental examples

Before presenting formal evaluations, we first give experimental examples over the above experimental setup to show the applicability of our approach. Consider the query Q_0 : “finding *recent* publications *highly* cited *written by* Rakesh Agrawal, and giving **twice** weight as much on recentness as on citation.” Tables 1 and 2 show the top-15 search results for Q_0 with the aggregation functions **averaged sum** or **min**, respectively. Here, the *#cited* field denotes the cited times of a publication, and f_{Λ} is the *weighting function*. The search results are non-increasingly ordered according to the value of f_{Λ} calculated by Eq. (5). Let us consider the ranked 1st entry in Table 1. Since $\text{Recent}(2002) = \frac{2002-1990}{2003-1990} = 0.9230$, $\text{Highly}(5) = 1 - \exp(-\frac{5}{30}) = 0.1535$ and the weighting $\Lambda = (\lambda_1 = \frac{2}{3}, \lambda_2 = \frac{1}{3})$, we get $f_{\Lambda} = (\frac{2}{3} - \frac{1}{3}) \cdot 0.9230 + 2 \cdot \frac{1}{3} \cdot \frac{0.9230 + 0.1535}{2} = 0.6665$. The same holds for those results in Table 2, except that the aggregation function **averaged sum** used in Table 1 is replaced by **min**.

All results in the two tables satisfy the query Q_0 with a degree of relevance, however, we can still find an important difference between Tables 1 and 2, that is, the ranked results in Table 1 are more discriminative compared to those in Table 2. This is because that, the standard min based function may lead to insufficiently discriminative ranking results since it only draws upon the comparison of the least satisfied properties. For example, the ranked 1st entry in Table 1 is only ranked 9th in Table 2 as its cited times is just 5, although it was published very recently (note: the present year $b = 2003$ in the experiments). The reason also explains the phenomenon that those ranked 9th, 10th and 12th entries in Table 1 are kicked out of Table 2. However, it does not mean that the search quality with **averaged sum** might be better than that with **min** or with other aggregation functions (e.g., **product**) in every application domain, since the selection of aggregation function f is application specific [1,45]. For the rest of the experiments, **averaged sum** is used to interpret f for our proposed weighting function f_{Λ} in Eq. (5).

4.3. Results and discussion

As far as we know, there are no published works compared to ours. Although some works have ever mentioned that ranking is necessary for best answers of fuzzy queries (e.g., [27]) or personal-

Table 1
Top-15 search results for query Q_0 with aggregation function averaged sum

Rank	Publication (title)	Year	#Cited	Value of f_λ
1	High-dimensional similarity joins	2002	5	0.6665
2	Mining quantitative association rules in large relational tables	1996	72	0.6107
3	On the computation of multidimensional aggregates	1996	55	0.5877
4	Mining association rules with item constraints	1997	31	0.5737
5	Modeling multidimensional databases	1997	31	0.5737
6	Mining sequential patterns	1995	84	0.5694
7	Mining sequential patterns: generalizations and performance improvements	1996	37	0.5439
8	SPRINT: a scalable parallel classifier for data mining	1996	37	0.5439
9	Fast algorithms for mining association rules	1994	222	0.5382
10	Privacy-preserving data mining	2000	2	0.5343
11	SLIQ: a fast scalable classifier for data mining	1996	32	0.5263
12	Mining association rules between sets of items in large databases	1993	246	0.4870
13	Efficient similarity search in sequence databases	1993	71	0.4559
14	Querying shapes of histories	1995	26	0.4496
15	Discovering trends in text databases	1997	9	0.4453

Table 2
Top-15 search results for query Q_0 with aggregation function min

Rank	Publication (title)	Year	#Cited	Value of f_λ
1	Mining association rules with item constraints	1997	31	0.5384
2	Modeling multidimensional databases	1997	31	0.5384
3	Mining quantitative association rules in large relational tables	1996	72	0.4615
4	On the computation of multidimensional aggregates	1996	55	0.4615
5	Mining sequential patterns: generalizations and performance improvements	1996	37	0.4615
6	SPRINT: a scalable parallel classifier for data mining	1996	37	0.4615
7	SLIQ: a fast scalable classifier for data mining	1996	32	0.4615
8	Parallel mining of association rules	1996	15	0.4161
9	High-dimensional similarity joins	2002	5	0.4100
10	Mining sequential patterns	1995	84	0.3846
11	Querying shapes of histories	1995	26	0.3846
12	Developing tightly-coupled data mining applications on a relational database system	1996	11	0.3584
13	Discovering trends in text databases	1997	9	0.3522
14	Fast similarity search in the presence of noise, scaling, and translation in time-series databases	1995	12	0.3479
15	A linear method for deviation detection in large databases	1996	10	0.3428

ized queries (e.g., [36]) over the semantic web, no evaluation results or further details are given in their works. However, in the experiments we still try to demonstrate the effect of our approach through comparing with some existing possible solutions: (1) **Baseline**: The query language is based on SPARQL with SQL-like exact boolean matching, and hence query results are ordered randomly; (2) **FNW**: This model enables fuzzy queries but no weighting assignment, and search results are ranked by *averaged sum*; and (3) **SFW**: This model supports fuzzy queries with weighting assignment as in ours. The only difference between SFW model and our approach is that, search results in the former are ranked by the simply assumed weighting function $f_\lambda = \min(\lambda_1 \cdot p_1(\mathbf{e}_1), \dots, \lambda_k \cdot p_k(\mathbf{e}_k))$ which is applied in [31].

We also evaluate different ranked results lists in agreement through giving small changes on the weights to show the continuity of our proposed weighting function f_λ .

4.3.1. Evaluation metrics

We perform the evaluations over popular retrieval metrics, namely *Normalized Discounted Cumulative Gain* (nDCG) [19], *Precision*, and a variant of *Kendall's τ* [9], as described below:

– *nDCG at K*: nDCG is a retrieval measure which not only estimates the number of relevant results in the ranked list, but also incurs a penalty for relevant results that are ranked low in the list [19]. [19] showed that nDCG gives more credit with high precision at top ranks than other evaluation measures, and so it is intensively used in TREC. For a given query Q , the ranked results are examined top-down, and nDCG is calculated as:

$$nDCG_Q = M_Q \sum_{i=1}^K \frac{2^{label(i)} - 1}{\log(1 + i)} \quad (6)$$

where M_Q is a normalization constant such that an ideal ordering would receive nDCG of 1; and $label(i)$ is an integer relevance label from 0 to 5 (e.g., 0="irrelevant", 1="slightly relevant", 5="perfectly relevant", etc.) of the result at position i in the top- K results. This metric is mainly used to compare the quality of our approach with three other models.

– *Precision at K*: Precision is a traditional standard retrieval metric. Precision at K gives the fraction of results in the top- K results that are relevant. The relevance value is regulated as that in nDCG, namely from 0 to 5. Then we average relevance values for all test queries. Hence, this metric measures user satisfaction with the top- K results on average.

– *Kendall's τ* : Kendall's τ is a popular distance measure between two ranked lists through counting the number of pair-wise disagreements between two lists [9]. We use a variant of *Kendall's τ* as similarity measurement which measures the degree to which the relative orderings of the top- K entries of two ranked lists are in agreement. Given two top- K lists τ_1 and τ_2 , let U be the union set of distinct elements in τ_1 or τ_2 . Let τ'_1 contains $U - \tau_1$ which appears after all the elements in τ_1 . The τ'_2 is defined similarly. Then we define the similarity measure as follows:

$$S(\tau_1, \tau_2) = \frac{|(u, v) : \tau'_1(u) < \tau'_1(v), \tau'_2(u) < \tau'_2(v)|}{(|U - 1|)|U|/2} \quad (7)$$

where $S(\tau_1, \tau_2)$ is in the range $[0, 1]$, and the larger value indicates τ_1 is more similar to τ_2 . For $u, v \in U$, if $\tau'_1(u) < \tau'_1(v)$, we say u is

ahead of v in τ_1 . The same holds for $\tau_2(u) < \tau_2(v)$. This metric is used to evaluate different ranked lists in agreement when small changes are made on the weights.

4.3.2. Comparison of nDCG at K

Since queries supported in our approach can be arbitrarily complex and should rely on the ontology schema, it is almost impossible to randomly generate a large number of legal queries. Hence, we manually prepared 20 test queries for the evaluation. We also assigned a numerical weighting for each test query. Then, we separately sent each query to our model and the other three models, such that there were four different result lists returned for each query. To judge the relevance of search results, we resorted to human participation. The human based evaluation is simple and subjective, however, it still can indicate the effect of our approach compared with other models in a certain degree. We invited 10 CS students who are Ph.D. or master candidates. Among them, three students are from HK CityU and the others are from our lab of HUST. Each student explicitly labeled the top- K ($K = 15$, occasionally less than 15) search results of every list using a six point scale ranging from 5 down to 0 (i.e., 5=“perfectly relevant”, and 0= “irrelevant”). We averaged 10 scores labeled by students for every result entry at position i as the relevance label $label(i)$ in Eq. (6).

We selected five test queries among those 20 and discuss their evaluation results on nDCG. Table 3 lists the selected queries with

weighting assignments. Fig. 4 shows the results of performance comparison on nDCG. Particularly, Fig. 4a–e give comparisons on nDCG@K of five selected queries over four different models. From Fig. 4a, we can see that our proposed model outperforms Baseline, FNW and SFW with respect to nDCG of Q_1 . The nDCG of Baseline is the lowest among four models. This is because the query language in Baseline is SPARQL and its results are only ordered randomly. The FNW model does not consider weighting, however, its search results are ranked by *averaged sum* such that better than Baseline model. The assumed weighting function $f_\Lambda = \min(\lambda_1 \cdot p_1(\mathbf{e}_1), \dots, \lambda_k \cdot p_k(\mathbf{e}_k))$ for ranking results is applied in SFW. Since min based function may lead to insufficiently discriminative ranking results, the search quality of our model excels almost 15% over that of SFW. In particular, we can observe that our model keeps nDCG of 1 for the several top results (i.e., top 5). This is critical for searching over large scale datasets since users usually pay more attention on the top results. For Q_3 in Fig. 4c, we can find that nDCG of Baseline is about 0.25 on average, which is even lower than that of Q_1 in Fig. 4a (about 0.50) and of Q_2 in Fig. 4b (about 0.45). The reason is that the keyword “database” in Q_3 is very popular in our datasets such that many results with low fuzzy predicate *IsAbout* values are returned by Baseline without ordering. Regarding to Q_5 in Fig. 4e, we observe that nDCG of FNW is close to that of our model. For the weighting $\Lambda = (\lambda_1 = 0.4, \lambda_2 = 0.3, \lambda_3 = 0.3)$ on Q_5 , the weights λ_1, λ_2 and λ_3 are very close. Recall the first property of f_Λ in Section 3.5. That property indicates that, if all of the weights in Λ are equal, then the weighting function f_Λ should be equal to the aggregation

Table 3

Queries selected from test set

ID	User Query	Weighting (Λ)
Q_1	recent publications highly cited written by Rakesh Agrawal	(2/3, 1/3)
Q_2	recent publications highly cited written by Rakesh Agrawal	(1/3, 2/3)
Q_3	highly cited publications talking about database	(2/3, 1/3)
Q_4	recent publications highly cited talking about knowledge representation	(0.4, 0.2, 0.4)
Q_5	recent publications highly cited talking about knowledge representation	(0.4, 0.3, 0.3)

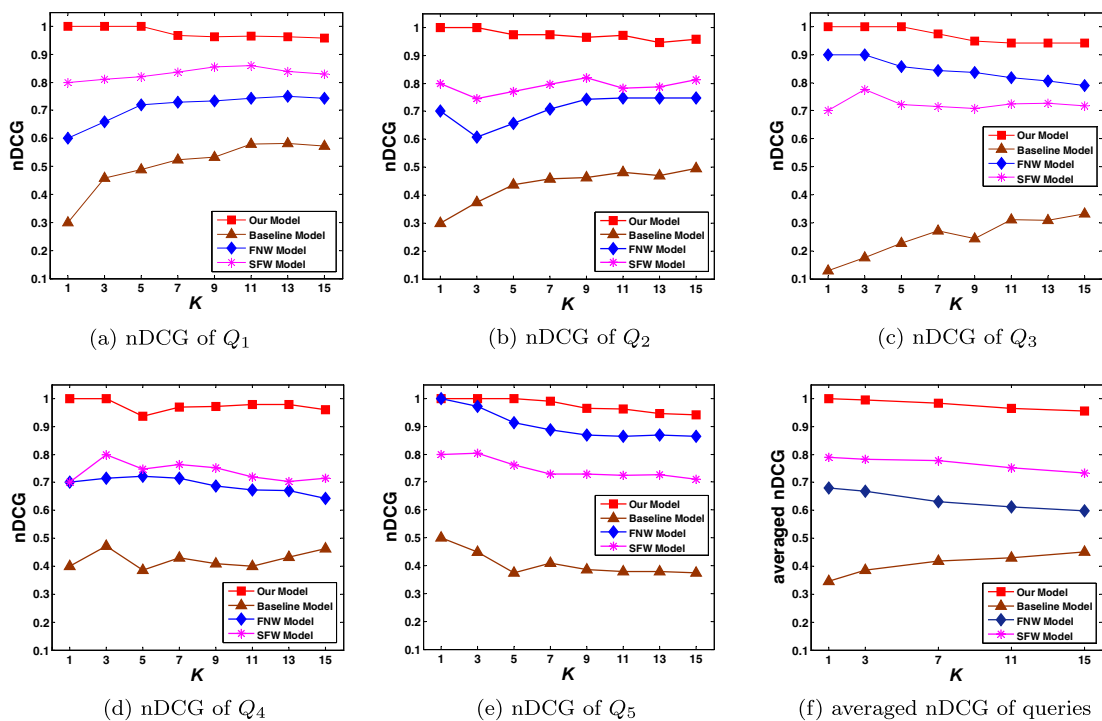


Fig. 4. Evaluations of nDCG at K for queries in our model, Baseline, FNW and SFW when varying K.

function f which does not consider weights. Therefore, if weights are all equal, our model degenerates to FNW. The nDCG is computed for each of 20 test queries, and then is averaged. The final averaged nDCG is presented in Fig. 4f which indicates that our approach outperforms other models.

4.3.3. Comparison of precision at K

Fig. 5 reports averaged Precision at K for the test queries set. For a given query, Precision at K gives the fraction of results in the top- K results that are relevant to the query. The relevance value is regulated as in nDCG, i.e., from 0 to 5. Then we averaged relevance values for 20 test queries. Fig. 5 shows that our model can best satisfy user search intension with the top K results on average. In particular, our model almost has averaged precision@1 of 100%, compared to 34% of Baseline, 68% of FNW, and 79% of SFW. This is very useful when searching over large scale semantic web data.

4.3.4. Effect of small changes on weighting

To evaluate the continuity of our proposed weighting function f_λ , we impose small changes on weights and measure two ranked lists τ_0 and τ_i in agreement by computing similarity measure $S(\tau_0, \tau_i)$. Let us take the query Q_4 in Table 3 as an example. The original weighting on Q_4 is $\Lambda = (\lambda_1 = 0.4, \lambda_2 = 0.2, \lambda_3 = 0.4)$. The ranked list is τ_0 . Then we change the weighting as in Table 4. Accordingly, we have ranked lists $\tau_i, 1 \leq i \leq 6$. The length of ranked lists is limited to top $k = 100$. Fig. 6 shows the similarity measure results. From the figure, we can see that similarity measure $S(\tau_0, \tau_i)$ has the largest value of 1 when $i = 0$. That is certain since the top- K lists $\tau_0 \equiv \tau_0$. The similarity measures $S(\tau_0, \tau_1)$ and $S(\tau_0, \tau_4)$ have high values 0.955 and 0.971, respectively, when very small changes ($\lambda_1 \pm 1\%$) are given on weighting. Therefore, the two ranked lists τ_0 and τ_1 (τ_4) have high agreement in the relative orderings of the top- K results. When changes ($\lambda_1 \pm 5\%, \lambda_2 \pm 5\%, \lambda_3 \pm 10\%$) are given on weighting, $S(\tau_0, \tau_3)$ and $S(\tau_0, \tau_6)$ have values 0.793 and 0.815, respectively. From Fig. 6, we can see that the weighting function f_λ has the good property of continuity.

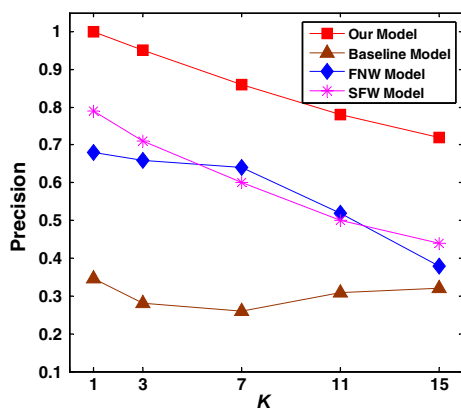


Fig. 5. Evaluation of Precision at K .

Table 4

Small changes of weights on fuzzy predicates in query Q_4

Ranked list	Weighting $\Lambda = (\lambda_1, \lambda_2, \lambda_3)$		
	λ_1 (%)	λ_2 (%)	λ_3 (%)
τ_0	0.4	0.2	0.4
τ_1	0.4 + 1	0.2 + 1	0.4 - 2
τ_2	0.4 + 3	0.2 + 3	0.4 - 6
τ_3	0.4 + 5	0.2 + 5	0.4 - 10
τ_4	0.4 - 1	0.2 - 1	0.4 + 2
τ_5	0.4 - 3	0.2 - 3	0.4 + 6
τ_6	0.4 - 5	0.2 - 5	0.4 + 10

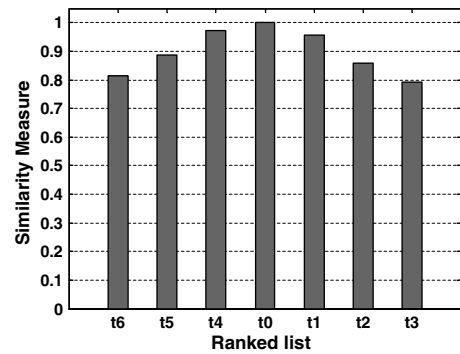


Fig. 6. Similarity measure $S(\tau_0, \tau_i)$ between ranked lists τ_0 and $\tau_i, 1 \leq i \leq 6$.

5. Conclusion and future work

In this paper, we propose a novel approach which formulates one's search request through tightly combining fuzziness together with the user's subjective weighting importance over multiple search properties. In addition, a ranking mechanism based on the weighed fuzzy query representation is defined. The ranking method generates a set of "degree of relevance" – an overall score which reflects both fuzzy predicates and the user's personalized preferences in the search request. Therefore, search results shall be properly ordered in terms of their "degree of relevance" with respect to best matching the search intension. The experimental results show that our approach is feasible. Additionally, our method outperforms Baseline, FNW and SFW models approximately by 60%, 35% and 20% with respect to nDCG. The experiments also show significant averaged precision improvements over other models. In particular, our method has substantial performance improvements for the several top results, which is critical for searching over large scale semantic web data.

Future research includes several directions. First, our current solution of weighting assignment relies on users explicitly assigning numerical weights to properties through the query interface, and hence imposes some cognitive overhead on users. We are exploring methods to automatically assign weights through relevance feedback strategy and predicting users preferences. Another promising direction for our work is to incorporate our approach into rules thus able to support more powerful reasoning based intelligent semantic web search.

Acknowledgements

This work is supported by the National 973 Key Basic Research Program under Grant No. 2003CB317003, and the Cultivation Fund of the Key Scientific and Technical Innovation Project, Ministry of Education of China under Grant No. 705034, and also the CityU Strategic Research grant 7002102 and 7002214.

References

- [1] D. Aksoy, M. Franklin, RxW: a scheduling approach for large-scale on-demand data broadcast, IEEE/ACM Transactions on Networking 7 (6) (1999) 846–880.
- [2] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider (Eds.), The Description Logic Handbook: Theory, Implementation, and Applications, Cambridge University Press, 2003.
- [3] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, L.A. Stein, OWL Web Ontology Language Reference. <<http://www.w3.org/TR/owl-ref/>>, 2007 (accessed July).
- [4] T. Berners-Lee, J. Hendler, O. Lassila, The semantic web, Scientific American 284 (5) (2001) 34–43.
- [5] M. Berry, Z. Drmac, E. Jessup, Matrices, vector spaces, and information retrieval, SIAM Review 41 (2) (1999) 335–362.
- [6] P. Bosc, O. Pivert, SQLf: a relational database language for fuzzy querying, IEEE Transactions on Fuzzy Systems 3 (1) (1995) 1–17.

- [7] H. Chen, S. Fellah, Y. Bishr, Rules for geospatial semantic web applications, in: Proceedings of the W3C Workshop on Rule Languages for Interoperability, Washington, April 2005.
- [8] F. Crestani, Application of spreading activation techniques in information retrieval, *Artificial Intelligence Review* 11 (6) (1997) 453–482.
- [9] C. Dwork, R. Kumar, M. Naor, D. Sivakumar, Rank aggregation methods for the web, in: Proceedings of the 10th International Conference on World Wide Web (WWW), Hong Kong, 2001, pp. 613–622.
- [10] FaCT++. <<http://owl.man.ac.uk/factplusplus/>>, 2007 (accessed July).
- [11] R. Fagin, E.L. Wimmers, Incorporating user preferences in multimedia queries, in: Proceedings of the Sixth International Conference on Database Theory (ICDT), 1997, pp. 247–261.
- [12] A. Ferrara, L.A. Ludovico, S. Montanelli, S. Castano, G. Haus, A semantic web ontology for context-based classification and retrieval of music resources, *ACM Transactions on Multimedia Computing, Communications and Applications* 2 (3) (2006) 177–198.
- [13] C. Golbreich, O. Bierlaire, O. Dameron, B. Gibaud, Use case: ontology with rules for identifying brain anatomical structures, in: Proceedings of the W3C Workshop on Rule Languages for Interoperability, Washington, April 2005.
- [14] B.N. Groszof, I. Horrocks, R. Volz, S. Decker, Description logic programs: combining logic programs with description logic, in: Proceedings of the 12th International Conference on World Wide Web (WWW), Budapest, Hungary, 2003, pp. 48–57.
- [15] R. Guha, R. McCool, E. Miller, Semantic search, in: Proceedings of the 12th International Conference on World Wide Web (WWW), Budapest, Hungary, 2003, pp. 700–709.
- [16] E. Herrera-Viedma, Modeling the retrieval process for an information retrieval system using an ordinal fuzzy linguistic approach, *Journal of the American Society for Information Science and Technology* 52 (6) (2001) 460–475.
- [17] I. Horrocks, P.F. Patel-Schneider, S. Bechhofer, D. Tsarkov, OWL rules: a proposal and prototype implementation, *Journal of Web Semantics* 3 (1) (2005) 23–40.
- [18] I. Horrocks, U. Sattler, S. Tobies, Practical reasoning for very expressive description logics, *Journal of the Interest Group in Pure and Applied Logics* 8 (3) (2000) 293–323.
- [19] K. Jarvelin, J. Kekalainen, Cumulated gain-based evaluation of IR techniques, *ACM Transactions on Information Systems* 20 (4) (2002) 422–446.
- [20] Jena. <<http://jena.sourceforge.net/>>, 2007 (accessed July).
- [21] KAON2. <<http://kaon2.semanticweb.org/>>, 2007 (accessed July).
- [22] G. Koutrika, Y. Ioannidis, Personalization of queries in database systems, in: Proceedings of the 20th International Conference on Data Engineering (ICDE), 2004, pp. 597–608.
- [23] C. Li, K.C. Chang, I.F. Ilyas, S. Song, RankSQL: query algebra and optimization for relational top-*k* queries, in: Proceedings of the 24th ACM SIGMOD International Conference on Management of Data (SIGMOD), 2005, pp. 131–142.
- [24] L. Li, I. Horrocks, A software framework for matchmaking based on semantic web technology, in: Proceedings of the 12th International Conference on World Wide Web (WWW), Budapest, Hungary, 2003, pp. 331–339.
- [25] Lucene. <<http://lucene.apache.org/>>, 2007 (accessed July).
- [26] C. Lutz, Description logics with concrete domains – a survey, *Advances in Modal Logic*, vol. 4, King's College Publications, 2003.
- [27] C. Meghini, F. Sebastiani, U. Straccia, A model of multimedia information retrieval, *Journal of the ACM* 48 (5) (2001) 909–970.
- [28] B. Motik, U. Sattler, R. Studer, Query answering for OWL-DL with rules, in: Proceedings of the Third International Semantic Web Conference (ISWC), Hiroshima, Japan, 2004, pp. 549–563.
- [29] X. Ning, H. Jin, H. Wu, RSS: a framework enabling ranked search on the semantic web, *Information Processing and Management* 44 (2) (2008) 893–909.
- [30] J.Z. Pan, I. Horrocks, Web ontology reasoning with datatype groups, in: Proceedings of the Second International Semantic Web Conference (ISWC), 2003, pp. 47–63.
- [31] J.Z. Pan, G. Stoilos, G. Stamou, V. Tzouvaras, I. Horrocks, f-SWRL: a fuzzy extension of SWRL, *Journal of Data Semantics VI* (2006) 28–46.
- [32] Protégé. <<http://protege.stanford.edu/>>, 2007 (accessed July).
- [33] RacerPro. <<http://www.racer-systems.com/>>, 2007 (accessed July).
- [34] C. Rocha, D. Schwabe, M. Aragao, A hybrid approach for searching in the semantic web, in: Proceedings of the 13rd International Conference on World Wide Web (WWW), New York, USA, 2004, pp. 374–383.
- [35] M. Schmidt-Schauß, G. Smolka, Attributive concept descriptions with complements, *Artificial Intelligence* 48 (1) (1991) 1–26.
- [36] W. Siberski, J.Z. Pan, U. Thaden, Querying the semantic web with preferences, in: Proceedings of the Fifth International Semantic Web Conference (ISWC), 2006, pp. 612–624.
- [37] SPARQL query language for RDF. <<http://www.w3.org/TR/rdf-sparql-query/>>, 2007 (accessed July).
- [38] G. Stoilos, N. Simou, G. Stamou, S. Kollias, Uncertainty and the semantic web, *IEEE Intelligent Systems* 21 (5) (2006) 84–87.
- [39] U. Straccia, Fuzzy \mathcal{ALC} with fuzzy concrete domains, in: Proceedings of the International Workshop on Description Logics (DL), 2005.
- [40] U. Straccia, Reasoning within fuzzy description logics, *Journal of Artificial Intelligence Research* 14 (1) (2001) 137–166.
- [41] D. Vallet, P. Castells, M. Fernandez, P. Mylonas, Y. Avrithis, Personalized content retrieval in context using ontological knowledge, *IEEE Transactions on Circuits and Systems for Video Technology* 17 (3) (2007) 336–346.
- [42] L.A. Zadeh, A note on web intelligence, world knowledge and fuzzy logic, *Data and Knowledge Engineering* 50 (3) (2004) 291–304.
- [43] L.A. Zadeh, Fuzzy sets, *Information and Control* 8 (3) (1965) 338–353.
- [44] L. Zhang, Y. Yu, J. Zhou, C. Lin, Y. Yang, An enhanced model for searching in semantic portals, in: Proceedings of the 14th International World Wide Web Conference (WWW), 2005, pp. 453–462.
- [45] H.J. Zimmermann, *Fuzzy Set Theory*, third ed., Kluwer Academic Publishers, Boston, 1996.