A Fuzzy Logic-based Approach to Semantic Query Answering with Missing Values

Fernando Bobillo, Carlos Bobed, Eduardo Mena
Aragon Institute of Engineering Research (I3A)
University of Zaragoza, Zaragoza, Spain
{fbobillo,cbobed,emena}@unizar.es

Umberto Straccia
CNR-ISTI
Pisa, Italy
umberto.straccia@isti.cnr.it

2024 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE) | 979-8-3503-1954-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/FUZZ-IEEE60900.2024.10611869

***Abstract*—Semantic Query Answering consists of retrieving individuals from a Knowledge Base, usually an OWL 2 ontology or an RDF knowledge graph, that satisfy a Semantic Query expressed via query terms. This paper proposes a novel approach to improve Semantic Query Answering in cases where individuals have missing values with respect to the query terms occurring in a Semantic Query. In our approach, the retrieved instances satisfy some query terms, but not necessarily all. Our general approach, based on the use of fuzzy aggregation operators, is complemented with concrete strategies to evaluate such Semantic Queries, together with an implemented prototype.**

***Index Terms*—ontologies, semantic queries, query relaxation, fuzzy aggregation, OWA**

I. Introduction

One of the reasons that explain the success of many Artificial Intelligence applications is the exploitation of knowledge. Modern intelligent applications represent knowledge and reason about it using typically *Semantic Web Languages* (SWL) such as OWL 2 and RDF knowledge graphs [1], [2]. Both OWL and RDF are formal and shared specifications of the vocabulary and data of a domain of interest, also called, in broad terms, *Knowledge Base* (KB). Using SWL has many advantages, proving easier interoperability, knowledge maintenance, information integration, reuse, and knowledge discovery.

In this paper, we will focus on the problem of Semantic Query Answering (SQA) which consists of retrieving individuals from a KB that satisfy a *Semantic Query* (SQ) expressed via *Query Terms* (QTs). In particular, we propose a novel approach to improve SQA in scenarios with incomplete information, where individuals have missing values with respect to the QTs occurring in a SQ. The idea is that the answer includes individuals where some, but not necessarily all QTs, are satisfied.

Please note that in SQA the so-called *Open World Assumption* holds, that is, one assumes that an agent does not have complete knowledge about a world and, thus, it only makes deductions that follow logically from statements that are known to be true. In particular, the absence of information does not entail its falseness. This latter case is what is assumed in the so-called *Closed World Assumption* instead [3].

Under the Open World Assumption, very often, user queries are too restrictive and may not provide a satisfactory result. A revision of the query is desired to improve the result. Query relaxation consists of loosening some QTs included in a user query in order to increase the number of retrieved individuals. For instance, if we want to buy a house, it is almost certain that we will have to relax some of the requirements that the house of our dreams would need to satisfy (such as, affordable price, enlarged rooms, beautiful views), as illustrated by the following running example (in a different domain).

Example 1 (Running example)**.** *Assume we have a KB about beers, as illustrated in the table below (empty cells denote an unknown value):*

Beer	*Lager*	*Color*	*Industrial*	*Alcohol*
b_1	*true*	*Golden*	*true*	4.5
b_2	*true*	*Golden*		4.6
b_3	*true*			4.6
b_4				
b_5	*false*	*Amber*	*false*	4.7
b_6	*false*	*Dark*	*false*	4.8
b_7		*Dark*	*false*	

Suppose now that a user is looking for industrial Golden lager beers. Such a request may be expressed informally (see Example 3 for the formal definition) by the query

$$Q = \mathsf{Lager} \wedge \mathsf{Color}(\mathsf{Golden}) \wedge \mathsf{Industrial} \qquad (1)$$

with three QTs: Lager, $\mathsf{Color}(\mathsf{Golden})$, *and* $\mathsf{Industrial}$.

Using traditional approaches, only beer b_1 would be retrieved. However, although we are sure that b_5, b_6 and b_7 do not satisfy the query, we are not sure whether b_2, b_3, and b_4 satisfy it or not, as we do not know: 1) if they are industrial or not, 2) the beer style of b_4 (is it Lager or not?), and 3) the colors of b_3 and b_4. The Open World Assumption would completely discard b_2, b_3, and b_4. However, a user may still be interested in retrieving them, though likely with a 'lower score' than b_1. □

Traditional QA approaches [4] focus on increasing the number of retrieved individuals when the query is very specific and has few results. They are pessimistic in the sense that if there is no guarantee that those individuals meet all the QTs (viz. restrictions), then they are not retrieved. In our case, they will be retrieved with a lower score as long as there is no evidence that they do not meet some QT. In particular, we assign weights to the different QTs and, to combine them,

we consider the use of different *Aggregation Operators* (AOs) widely used in fuzzy logic. Moreover, we aim at being able not only to relax the query, but to also assess the importance of each relaxation under Open World Assumption in KBs with potentially many unknown values.

In summary, the main contributions of our paper are the following ones:

- we provide a novel approach to SQA based on fuzzy aggregation which is suitable for scenarios with missing values (incomplete information);
- we discuss several ways to compute the weights of the different QTs in the query in an automatic way; and
- we implement our approach in a prototype, and discuss and illustrate a use case.

The remainder of this paper is structured as follows. Section II provides the necessary background. Section III details our novel proposal for semantic query relaxation using aggregation, and Section IV addresses how to compute the weights. Next, Section V discusses some properties of the aggregation operator and Section VI describes an implementation and an illustrative use case. Finally, Section VII compares our approach with related work and Section VIII summarizes the main contributions and points out some ideas for future work.

II. Background

This section provides the minimal background on SWLs (Section II-A) and *Aggregation Operators* (AOs) (Section II-B), required to follow the rest of the paper.

A. Semantic Web Languages

The Semantic Web architecture proposes languages to represent knowledge via so-called *ontologies* [5]. An ontology offers a common vocabulary, terminology, a conceptual model, and a formal semantics which makes it possible to perform semantic reasoning. Ontology advantages include promoting knowledge sharing and reuse, being easy to read and understand for machines (software applications), and providing a separation of domain knowledge from the application level, which is particularly useful for intelligent systems [1].

A current standard is the *Web Ontology Language OWL* 2 [6]. The main ingredients of OWL ontologies to model a domain are the following ones:

- *Instances/individuals* are particular elements of the domain, such as duff beer or duffBrewery[1].
- *Classes*, also known as *concepts*, are sets of instances, such as Beer or Brewery. Classes are typically organised into a class hierarchy (subclass/superclass) or taxonomy. For example, Lager is a subclass of Beer.
- *Datatypes/concrete domains* are elements that belong to a domain that is already known to the machine, such as reals, integers, strings, dates, etc.
- *Properties/roles* are binary relationships between a pair of elements. *Object properties* relate a pair of individuals, such as brewedBy, while *data properties* link an individual with a datatype value (of a concrete datatype), such as price or alcohol, with a numerical range.
- *Axioms* are constraints that every ontology element must satisfy, such as the membership of an individual to a class.

Starting from atomic classes, complex classes, i.e., complex concepts, are then defined inductively, depending on the allowed operators.[2] For example, in OWL 2, given classes C_a and C_b, it is possible to define their Boolean combinations: *negation/complement* of C_a (denoted $\neg C_a$), *conjunction/intersection* of C_a and C_b, and *disjunction/union* of C_a and C_b.

The type of allowed *axioms* also depend on the expressivity of the ontology language at hand. In this paper, the following ones will be relevant:

- Concept assertions state that an individual belongs to a concept, such as Brewery(duffBrewery).
- Object property assertions state that two individuals are related via an object property, such as brewedBy(duff, duffBrewery). Negated object property assertions state that a relation between two individuals does not hold.
- Data property assertions relate an individual and a datatype value via a data property, such as price(duff, 5.0). Negated data property assertions state the absence of a relation.
- Class disjointness state that two or more classes cannot share instances. Similarly, property disjointness state that two or more properties cannot link the same pair of individuals.
- Individual equality (resp. inequality) state that two or more individuals must be interpreted as the same entity (resp. different entities). They are necessary because there is no Unique Name Assumption.
- Property functionality states that an individual cannot have more than a value for a given property.

OWL 2 ontologies have a formal, logical counterpart given by *Description Logics* [7]. Defining the semantics of ontologies is out of the scope of this paper. Nevertheless, we will use the following definition: an ontology $\mathcal{O}$ *entails* an axiom τ, or also τ is a *logical consequence* of $\mathcal{O}$, (denoted $\mathcal{O} \models \tau$) iff τ is true in every logical model of $\mathcal{O}$ (see also [8]).

Let us also note that OWL 2 is designed on top of other essential technologies in the Semantic Web stack, such as the eXtensible Markup Language (XML), the *Resource Description Framework* (RDF), and *RDF Schema* (RDFS):

- *XML* allows user to create their own tags and define document structure, but it does not support semantics.
- *RDF* is a standard model for data interchange [9] based on a triple representation of the form (s, p, o) where s is a *subject* (a resource), o an *object* (a resource or literal) and p is a binary *predicate* (a property). For example, $\langle$Duff, brewedBy, DuffBrewery$\rangle$ is a triple, whose intended meaning is brewedBy(Duff, DuffBrewery).

[1]For ease of presentation, we will consider the *Unique Name Assumption* (UNA), where different individuals denote different objects.

[2]OWL 2 provides also the specification of three sub-languages, called OWL 2 profiles, namely, OWL 2 QL, OWL 2 EL and OWL 2 RL, see https://www.w3.org/TR/owl2-profiles/.

- *RDFS* is an extension of RDF with a specific vocabulary to represent subclasses, subproperties, domain, and range restrictions [10].

Knowledge Graphs (KGs) are graphs representing real world data, whose nodes represent entities and whose edges represent relations between pairs of entities [2]. The most popular type of KGs are RDF graphs, which are labeled directed graphs composed of RDF triples. An RDF triple $\langle s, p, o\rangle$ links nodes s and o via an edge labelled as p. KGs usually come along with OWL 2 ontologies, which contain schema knowledge about the domain that the KG is about.

B. Aggregation Operators

Aggregation Operators (AOs) are mathematical operations that are employed to combine different pieces of information [11]. Given a domain $\mathbb{D}$, an AO is a mapping $@ : \mathbb{D}^K \to \mathbb{D}$, aggregating K values $x_1, \ldots, x_K$ into a single one. Usually, $\mathbb{D} = [0, 1]$. Some typical examples are the average, the weighted mean, the maximum, or the minimum. AOs often assume a vector of weights $W = [w_1, \ldots, w_K]$ such that $w_i \in [0, 1]$ and $\sum_{i=1}^{K} w_i = 1$.

AOs are very important in fuzzy systems, as they provide an alternative way to combine information, different than the regular t-norm and t-conorm functions, used to define the conjunction and disjunction operations [12], [13].

Examples of AOs are the following ones. The *weighted mean* is defined as:

$$\mathbf{WMEAN}([w_1, \ldots, w_K], [x_1, \ldots, x_K]) = \sum_{i=1}^{K} w_i x_i \ . \quad (2)$$

Another popular example is the *Ordered Weighted Averaging* (OWA) operator [14]

$$\mathbf{OWA}([w_1, \ldots, w_K], [x_1, \ldots, x_K]) = \sum_{i=1}^{K} w_i x_{\sigma(i)} \ , \quad (3)$$

where $\sigma(i)$ is a permutation such that $x_{\sigma(1)} \geq x_{\sigma(2)} \geq \cdots \geq x_{\sigma(K)}$, i.e., $x_{\sigma(i)}$ is the i-th largest of the values $x_1, \ldots, x_K$ to be aggregated.

III. Query Relaxation using Aggregation

In some applications, it may be interesting to retrieve, with a lower score, individuals that only partially satisfy QTs, in the sense that some QT are not satisfied due to missing data (unknown values). In this section, we will show how to apply AOs to query relaxation over KBs in case of missing values, as the following Example 2 illustrates.

Example 2 (Running example cont.)**.** *Let us consider again Example 1 and the query (1) composed by three QTs. Let us rank the beers $b_1, \ldots, b_4$, using OWA and the weighting vector $W = [0.5, 0.333, 0.167]$ for the three QTs:*

b_1: $\mathbf{OWA}([0.5, 0.333, 0.167], [1, 1, 1]) = w_1 \cdot x_{\sigma(1)} + w_2 \cdot x_{\sigma(2)} + w_3 \cdot x_{\sigma(3)} = 0.5 \cdot 1 + 0.333 \cdot 1 + 0.167 \cdot 1 = 1$

b_2: $\mathbf{OWA}([0.5, 0.333, 0.167], [1, 1, 0]) = 0.5 \cdot 1 + 0.333 \cdot 1 + 0.167 \cdot 0 = 0.833$

b_3: $\mathbf{OWA}([0.5, 0.333, 0.167], [1, 0, 0]) = 0.5 \cdot 1 + 0.333 \cdot 0 + 0.157 \cdot 0 = 0.5$

b_4: $\mathbf{OWA}([0.5, 0.333, 0.167], [0, 0, 0]) = 0.5 \cdot 0 + 0.333 \cdot 0 + 0.167 \cdot 0 = 0$

Therefore, the ranking of the non-zero scored answers to the query (1), in decreasing score order, is:

$$\langle b_1, 1.0\rangle, \langle b_2, 0.833\rangle, \langle b_3, 0.5\rangle \ . \quad \square$$

Now, let

$$Q(x) = R_1(x) \wedge \cdots \wedge R_k(x)$$

be a *Conjunctive Query* (CQ) over an ontology $\mathcal{O}$, where x is a variable (that should be substituted with an individual) and $R_i(x)$ are QTs to be satisfied by the answers, which can have one of the following form:

- the membership to a class C, denoted $C(x)$;
- the relationship with a value v via a data property P, denoted $P(x, v)$; or
- the relationship with an individual a via an object property P, denoted $P(x, a)$.

Note that QTs relate to concept or properties where the first argument is a variable and not a fixed individual. Moreover, QTs may be seen as unary predicates.

Example 3. *Consider Example 1. Lager and Industrial are classes, and Color and Alcohol are data properties. The conjunctive query expressing (1) may be*

$$Q(x) = \mathit{Lager}(x) \wedge \mathit{Color}(x, \mathit{Golden}) \wedge \mathit{Industrial}(x) \quad (4)$$

where the data in the table is encoded via the ontology:

$\mathcal{O} = \{$ $\mathit{Lager}(b_1)$, $\mathit{Color}(b_1, \text{“Golden”})$, $\mathit{Industrial}(b_1)$, $\mathit{Alcohol}(b_1, 4.5)$,
$\mathit{Lager}(b_2)$, $\mathit{Color}(b_2, \text{“Golden”})$, $\mathit{Alcohol}(b_2, 4.6)$,
$\mathit{Lager}(b_3)$, $\mathit{Alcohol}(b_3, 4.6)$,
$\neg\mathit{Lager}(b_5)$, $\mathit{Color}(b_5, \text{“Amber”})$, $\neg\mathit{Industrial}(b_5)$, $\mathit{Alcohol}(b_5, 4.7)$,
$\neg\mathit{Lager}(b_6)$, $\mathit{Color}(b_5, \text{“Dark”})$, $\neg\mathit{Industrial}(b_6)$, $\mathit{Alcohol}(b_6, 4.8)$,
$\mathit{Color}(b_7, \text{“Dark”})$, $\neg\mathit{Industrial}(b_7)$ $\}$. $\square$

Definition 1 (Satisfied restriction)**.** *Let $R(x)$ be a QT, $\mathcal{O}$ be an ontology and a be an individual. We say that*

1) *the truth of $R(a)$ is* true *wrt $\mathcal{O}$ iff $\mathcal{O} \models R(a)$;*
2) *the truth of $R(a)$ is* false *wrt $\mathcal{O}$ iff $\mathcal{O} \models (\neg R)(a)$;*
3) *the truth of $R(a)$ is* unknown *wrt $\mathcal{O}$ iff $\mathcal{O} \not\models R(a)$ and $\mathcal{O} \not\models (\neg R)(a)$.*

Note that in Definition 1, case 3 does not apply under the Closed World Assumption as either case 1 or case 2 hold.

In the following, if the ontology is clear from the context, we will not write it explicitly.

For a given query Q and an individual a, we will write $unsat(Q(a))$ to denote that there is a QT $R_i(x) \in Q$ such that the truth of $R_i(a)$ is false.

Definition 2 (Score of a query)**.** *Given an ontology $\mathcal{O}$, the score of an individual a wrt a query Q is defined as*

$$Q(a) = \begin{cases} -1 & \text{if } unsat(Q(a)) \\ @_W(R_1(a), \ldots, R_k(a)) & \text{otherwise,} \end{cases} \quad (5)$$

where $@_W$ is an AO, W is a vector of weights $W = [w_1, \ldots, w_k]$ and $R_i(x) = 1$ if the truth of $R_i(a)$ is true, and $R_i(x) = 0$ if the truth of $R_i(a)$ is unknown.

Note that in the above definition $R_i(a) \in \{0, 1\}$. This may further be generalized by adopting fuzzy ontologies [15], [16] and, thus, one may consider partial truth $R_i(a) \in [0, 1]$ instead.

From the definitions of the WMEAN and OWA operators, the following result is easy to prove:

Proposition 1. *Let Q be a query and a an individual such that $unsat(Q(a))$ does not hold. Assume $w_i > 0$ for all $i \in \{1, \ldots, k\}$, and let $@_W \in \{\mathbf{WMEAN}, \mathbf{OWA}\}$. Then, the following properties hold:*

1) *$Q(a) = 1$ iff $R_i(a) = 1$, for all $i \in \{1, \ldots, k\}$.*
2) *$Q(a) = 0$ iff $R_i(a) = 0$, for all $i \in \{1, \ldots, k\}$.*
3) *$Q(a) \in (0, 1)$ if there are some $i, j \in \{1, \ldots, k\}$ such that $R_i(a) = 1$ and $R_j(a) = 0$.*

Notice that, although we have built the above definitions for an ontology, they could be applied to a KG similarly.

IV. Aggregation Operator Construction

In order to apply Definition 2 in practice, the problem we have to face is, given a query, to find an appropriate AO operator $@_W$ and, in particular, to find an appropriate vector of weights $[w_1, \ldots, w_k]$. Of course, there always exists the option that weights are defined by the user a priori. The topic of this section is to discuss more automatic techniques.

A. Proportional OWA

Consider a query Q with k QTs. A first simple idea is to build a vector $N = [N_1, \ldots, N_k]$ which represents the proportion of individuals for which exactly i QTs' truth is known. From the vector N, we may build the vector of weights $W = [w_1, \ldots, w_k]$ as follows:

$$w_{k-i+1} = \frac{N_i}{\sum_{j=1}^{k} N_j} \ . \tag{6}$$

Note that $w_i \in [0, 1]$ and $\sum_{i=1}^{k} w_i = 1$ hold.

Example 4 (Running example cont.)**.** *Consider Example 3 and query Q as per Eq. 4. Then $N = [1/7, 2/7, 3/7] = [0.143, 0.286, 0.429]$. Indeed, there are seven individuals and Q has three QTs. For three individuals (b_1, b_5, and b_6), we know the truth of all QTs; for two individuals (b_2 and b_7) we know the truth of exactly two QTs, i.e., the truth of one QT is unknown; for one individual (b_3) we know the truth of exactly one QT, i.e., the truth is unknown for two QTs; and for one individual the truth of all QTs is unknown (b_4). Then, from N, we then define the following vector of weights:*

1) *$w_{3-1+1} = w_3 = {}^{N_1}/_{\sum_{i=1}^{k} N_i} = {}^{0.143}/_{0.858} = 0.167$*
2) *$w_{3-2+1} = w_2 = {}^{N_2}/_{\sum_{i=1}^{k} N_i} = {}^{0.286}/_{0.858} = 0.333$*
3) *$w_{3-3+1} = w_1 = {}^{N_3}/_{\sum_{i=1}^{k} N_i} = {}^{0.429}/_{0.858} = 0.5$*

Thus, $W = [0.5, 0.333, 0.167]$ (rounded to three decimal places).

Now, let us evaluate the score of the beers. Firstly, note that by definition $Q(b_5) = Q(b_6) = Q(b_7) = -1$, as they have at least one QT whose truth is false (e.g., they are known to be not industrial). By Proposition 1 we already know that the score of b_1 is 1.0 (case 1), while the score of b_4 is 0.0 (case 2). It remains to address case 3 of Proposition 1.

Since the result depends on the number of QTs in Q whose truth is true, by case 3 of Proposition 1, it remains to consider the two cases, when the truth of exactly i QTs is unknown, with $i \in \{1, 2\}$. We will use two witness individuals x_1, x_2, where x_i is such that exactly the truth of i QTs is unknown. Then

1) *$Q(x_1) = Q(b_2)$; and*
2) *$Q(x_2) = Q(b_3)$.*

The scores can be found in Example 2. □

The next example illustrates more extreme cases.

Example 5. *Once more, consider Example 3 and query Q as per Eq. 4. However, assume now that for four individuals we have all the data, and, thus, the truth of all three QTs in Q is known; for two individuals the truth of one QT is unknown; and for one individual the truth of two QT is unknown. It can be verified that now $N = [1/7, 2/7, 4/7]$ and, thus, $W = [4/7, 2/7, 1/7] = [0.571, 0.286, 0.143]$. As in Example 4, we will only focus on case 3 of Proposition 1, considering the two cases of exactly i QTs' truth is unknown, with $i \in \{1, 2\}$. Again, we may use two witness individuals x_1, x_2, where x_i is such that for exactly i QTs' truth is unknown. Therefore,*

1) *$Q(x_1) = 0.571 \cdot 1 + 0.286 \cdot 1 + 0.143 \cdot 0 = 0.857$;*
2) *$Q(x_2) = 0.571 \cdot 1 + 0.286 \cdot 0 + 0.143 \cdot 0 = 0.571$.*

Let us consider now another case in which the amount of missing values in the ontology increases. Assume that for one individual we have all the data, and, thus, the truth of all three QTs in Q is known; for two individuals the truth of two QT is unknown, and for three individuals the truth of one QT is unknown. Now, $N = [3/7, 2/7, 1/7]$ and, thus, $W = [0.167, 0.333, 0.5]$. Therefore, the evaluation of the witness individuals is

1) *$Q(x_1) = 0.167 \cdot 1 + 0.333 \cdot 1 + 0.5 \cdot 0 = 0.5$*
2) *$Q(x_2) = 0.167 \cdot 1 + 0.333 \cdot 0 + 0.5 \cdot 0 = 0.167$*

Notice that the score lowers as more missing values occur in the ontology, e.g., we move from $Q(x_1) = 0.857$ to $Q(x_1) = 0.5$. □

Remark 1. *Alternatively, we could define the weights as*

$$w_i = \frac{N_i}{\sum_{i=1}^{k} N_i} \tag{7}$$

Equation 6 assigns a high score when there are few missing values and a low score when there are more of them, however in Equation 7 the more missing values, the higher score.

Let us consider again the former case in Example 5. From $N = [1/7, 2/7, 4/7]$ we get now by Eq. 7, $W = [0.143, 0.286, 0.571]$. Therefore, the score of the witness individuals is

- *$Q(x_1) = 0.143 \cdot 1 + 0.286 \cdot 1 + 0.571 \cdot 0 = 0.429$;*
- *$Q(x_2) = 0.143 \cdot 1 + 0.286 \cdot 0 + 0.571 \cdot 0 = 0.143$.*

Concerning the latter case in Example 5. From $N = [3/7, 2/7, 1/7]$ *we get now by Eq. 7,* $W = [0.5, 0.333, 0.167]$. *Therefore, the score of the witness individuals is*

1) $Q(x_1) = 0.5 \cdot 1 + 0.333 \cdot 1 + 0.167 \cdot 0 = 0.833$;
2) $Q(x_2) = 0.5 \cdot 1 + 0.333 \cdot 0 + 0.167 \cdot 0 = 0.5$.

Note that now the alternative definition assigns a lower score (0.429) *when there are fewer missing values, and a higher score* (0.833) *when there are more missing values. Selecting one option ore another one is a design choice.*

B. Weighted Mean and Distribution of the Values

Let us now discuss a more complex idea. Consider a query Q with k QTs $R_i(x)$, $i \in \{1, ..., k\}$. Let us define the *values* of R_i, denoted $val(R_i)$, in the following way:

- If $R_i(x) = C(x)$, where C is a class, then $val(R_i)$, is the multiset of *true* (resp. *false*) values for which there is an individual a such that the truth of $R_i(a)$ is true (resp. false).
- If $R_i(x) = P(x, v)$, where P is a property, then $val(R_i)$, is the multiset of values v' for which there is an individual a such that the truth of $P(a, v')$ is known.

With $sval(R_i)$ we denote the set representation of the multiset $val(R_i)$, and with I_{R_i} the set of individuals occurring in $\mathcal{O}$ for which the truth of $R_i(a)$ is known.

Now, we may consider the vector $V = [V_1, V_2, \ldots, V_k]$ where V_i is the proportion of individuals for which the truth of QT R_i is known (i.e., $|I_{R_i}|/|I_{\mathcal{O}}|$, where $I_{\mathcal{O}}$ is the set of individuals occurring in $\mathcal{O}$). For example, given a query with three QTs, $R_1(x), R_2(x), R_3(x)$, and 100 individuals, we may have $V = [90/100, 50/100, 10/100] = [0.9, 0.5, 0.1]$, which means informally that R_1 is very common, while $R_3(x)$ is very rare.

Interestingly, we can also take into account the distribution of the values inspired by information theory [17]. Specifically, we can consider a vector $H = [H_1, H_2, \ldots, H_k]$ where H_i denotes the *informativeness* of R_i, which we define as

$$H(R_i) = -\sum_{x \in sval(R_i)} p_i(x) \cdot \log_{10}\ p_i(x)\ , \tag{8}$$

where x is a value of R_i, and $p_i(x)$ is its probability, i.e.

$$p_i(x) = \frac{|val(R_i)(x)|}{|val(R_i)|}\ , \tag{9}$$

where $val(R_i)(x)$ is the restriction of $val(R_i)$ to the value x.[3] The higher $H(R_i)$, the more informative it is. $H(R_i)$ is inspired by Shannon's information entropy.

The next step is to build a vector of weights $W = [w_1, \ldots, w_k]$ from V and H. A possible solution is to define

$$\varpi_i = V_i \cdot H_i \tag{10}$$

and then normalize the values as

$$w_i = \frac{\varpi_i}{\sum_{i=1}^{k} \varpi_i}\ . \tag{11}$$

[3]Note that $val(R_i)(x)$ is a multiset of x values only.

Note that the weight is proportional to the number of defined values and to the informativeness, and that indeed $w_i \in [0, 1]$ and $\sum_{i=1}^{K} w_i = 1$.

Example 6 (Running example cont.)**.** *It can be shown that* $V = [5/7, 5/7, 4/7]$ *and* $H = [0.439, 0.577, 0.413]$ *(rounded to three decimal places), where the positions of the arrays correspond to* Lager, Color, *and* Industrial, *in this order. Therefore,* $W = [0.326, 0.429, 0.245]$.

Notice that, by definition, $Q(b_5) = Q(b_6) = Q(b_7) = -1$. *For the evaluation of the other beers we have*

- $Q(b_1) = 0.326 \cdot 1 + 0.429 \cdot 1 + 0.245 \cdot 1 = 1$
- $Q(b_2) = 0.326 \cdot 1 + 0.429 \cdot 1 + 0.245 \cdot 0 = 0.755$
- $Q(b_3) = 0.326 \cdot 1 + 0.429 \cdot 0 + 0.245 \cdot 0 = 0.326$
- $Q(b_4) = 0.326 \cdot 0 + 0.429 \cdot 0 + 0.245 \cdot 0 = 0$. □

Clearly, there exist other possible ways to normalize the ϖ_i values, such as *softmax normalization*, which slightly benefits elements with a higher weight:

$$w_i = \frac{e^{\varpi_i}}{\sum_{i=1}^{k} e^{\varpi_i}}\ . \tag{12}$$

However, it is obvious that if all values in $val(R_i)$ are equal, then $H(R_i) = 0$, so the weight w_i becomes 0 as well. To avoid this, in that case we may apply an idea based on Laplace smoothing:[4] for each R_i, we add to $val(R_i)$ a new value not occurring in $val(R_i)$, representing the *unknown* values. For example, if there are seven individuals with the same value, now the informativeness becomes

$$H(R_i) = -1/8 \cdot \log_{10}(1/8) - 7/8 \cdot \log_{10}(7/8) = 0.164\ .$$

Of course, this is not the only possible way to define the weights. Another possible solution is to define a linear combination

$$\varpi_i = \alpha \cdot V_i + (1 - \alpha) \cdot H_i\ , \tag{13}$$

where $\alpha \in [0, 1]$ is a constant that weights the relative importance of V_i and H_i in the final decision. In this case, even if $H(R_i) = 0$, $w_i \neq 0$ in general.

V. Some Properties

Now we will discuss some properties of the aggregation operators in our scenario. In particular, we will discuss their computational complexity, stability, behavior for different restriction types, behavior for the same number of satisfied restrictions, and the presence of zero weights.

A. Computational Complexity

A first observation is that, in weighted mean, some of the values ϖ can be reused, but the weights w_i must be recomputed, as changes in the ϖ require a new normalization.

Computing ϖ_i is computationally complex because it involves solving the instance retrieval problem for each concept C in the query and its negated concept $\neg C$, and retrieving

[4]Idea taken from the area of Natural Language Processing; basically, it consists in adding 1 to all the witnessed values of the ontology elements.

the property values for each property in the query and each individual in the system. Fortunately, we can precompute ϖ_i for each concept and property in the system. They could be stored as an OWL 2 annotation to each concept or property definition. Then, at query answering time, we can compute w_i taking into account only the values of ϖ_i associated to the concepts/properties in the query.

In OWA, this is not possible because N depends of the query, as it is based on the number of individuals that define the restrictions in the query.

B. Stability

The vectors of weights computed in the previous section depend on the query, as they depend on the QTs. It is therefore interesting to study how stable the previous approaches are when the query is slightly different.

Let us consider the case of less restrictive queries, i.e., given a query with some restrictions, we consider another query with a strict subset of the restrictions. As we will see in Example 7, in weighted mean the proportion between the weights remains the same; this is, however, not true for OWA.

Example 7. *Consider a less restrictive query than Example 1:*

$$Q_1(x) = \mathit{Lager}(x) \wedge \mathit{Color}(x, \mathit{Golden}) \ .$$

Now, b_7 is not known to falsify Q_1, i.e., $unsat(Q_1(b_7))$ does not hold.

Using OWA, we have $N = [2/7, 4/7] = [0.286, 0.571]$, so $W = [0.667, 0.333]$. Recall that in the original query (Example 4), $W = [0.5, 0.333, 0.167]$, so w_1 is 1.5 times higher than w_2. Now, w_1 is 2 times higher than w_2.

Using weighted mean, we have $V = [5/7, 5/7]$ and $H = [0.439, 0.577]$. The values ϖ_1, ϖ_2 are exactly the same as before, but we have to recompute the weights w_i by normalizing only these two ϖ values. More precisely, $W = [0.432, 0.568]$. Now, w_2 is 1.31 times higher than w_1, exactly as in the original query (Example 6).

The behavior of both approaches seems reasonable in the sense that for a less restrictive query (Q_1), individuals that satisfy all QTs but one (b_3) receive a higher degree. □

C. Informativeness and Restriction Types

Let us discuss an effect of the restriction types on the informativeness values. If a QT $R_i(x)$ consists of a concept membership, typically, there will be a lot of individuals with an unknown truth compared to the amount of values in $val(R_i)$, so the informativeness value will be low. Therefore, restrictions defined as concept memberships might tend to have a low weight.

Furthermore, the informativeness in real-valued data properties is expected to be higher than the informativeness of object properties. In general, the more possible values in $val(R_i)$, the higher the informativeness. Therefore, weights might tend to have a higher value.

D. Same Number of Satisfied Restrictions

We have mentioned above that in OWA the score depends on the number of satisfied restrictions, but not on which restrictions are satisfied QTs (so we focused on witness individuals x_i which satisfy i QTs). That does not happen in weighted mean, as the following example shows.

Example 8. *Let us consider the query:*

$$Q_2(x) = \mathit{Lager}(x) \wedge \mathit{Color}(x, \mathit{Dark}) \ .$$

We will compute the scores of b_3 and b_7, which are two individuals that are known to satisfy exactly one QT. As shown in Example 7, using OWA we have $W = [0.667, 0.333]$, and using weighted mean we have $W = [0.432, 0.568]$.
Now, using OWA, we have

- $Q_2(b_3) = \mathbf{OWA}([0.667, 0.333], [1, 0]) = 0.667 \cdot x_{\sigma(1)} + 0.333 \cdot x_{\sigma(2)} = 0.667 \cdot 1 + 0.333 \cdot 0 = 0.667$
- $Q_2(b_7) = \mathbf{OWA}([0.667, 0.333], [0, 1]) = 0.667 \cdot x_{\sigma(1)} + 0.333 \cdot x_{\sigma(2)} = 0.667 \cdot 1 + 0.333 \cdot 0 = 0.667$

Instead, using weighted mean, we have:

- $Q_2(b_3) = \mathbf{WMEAN}([0.432, 0.568], [1, 0]) = 0.432 \cdot 1 + 0.568 \cdot 0 = 0.432$
- $Q_2(b_7) = \mathbf{WMEAN}([0.432, 0.568], [0, 1]) = 0.432 \cdot 0 + 0.568 \cdot 1 = 0.568$ □

E. Zero Weights

Proposition 1 shows that the query score has some reasonable properties if weights are not zero. However, using our approach to build OWA weights, it is possible to get zero weights, which has some undesired effects, as the following example shows:

Example 9. *Let us now consider the query*

$$Q_3(x) = \mathit{Lager}(x) \wedge \mathit{Color}(x, \mathit{Golden}) \wedge \mathit{Alcohol}(x, \mathit{4.6}) \ .$$

Now, there are four individuals (b_1, b_2, b_5, b_6) with three QTs' truth known, one (b_3) with two QTs' truth known, zero individuals with one QTs' truth known, and two individuals (b_4, b_7) with zero QTs' truth known. Thus, $N = [0, 1/7, 4/7]$ and $W = [0.8, 0.2, 0]$. The scores of b_2 and b_3 are

- $Q_3(b_2) = \mathbf{OWA}([0.8, 0.2, 0], [1, 1, 1]) = 0.8 \cdot 1 + 0.2 \cdot 1 + 0 \cdot 1 = 1$
- $Q_3(b_3) = \mathbf{OWA}([0.8, 0.2, 0], [1, 0, 1]) = 0.8 \cdot 1 + 0.2 \cdot 1 + 0 \cdot 0 = 1$.

Notice that both beers receive the same score, although we know that beer b_1 satisfies all QTs, while b_2 does not: we do not know if b_2 has golden color or not. We would prefer to have scored b_1 higher than b_2, so this zero weight is undesired. □

VI. Implementation and Use Case

We have implemented a prototype tool implementing the approach discussed in the previous sections. Our prototype has been implemented in Java and uses *OWL API* 5.1[5] [18] and *HermiT* 1.3.8 [19] to process the ontologies. The inputs of the

[5] http://owlapi.sourceforge.net

prototype are the ontology and the lists of concepts, properties, and property values that characterize the SQ. The output is a ranked list of individuals for both aggregation operators. Individuals with rank 0 and -1 are also returned.

As a use case, we consider *Fuzzy Beer*, a fuzzy ontology with information about beers used in GimmeHop[6], a knowledge-based recommender system for mobile devices [20]. Fuzzy Beer ontology represents concepts (e.g., beer types or breweries), object properties (e.g., between beers and breweries), data properties (e.g., alcohol level, color, or bitterness), instances (e.g., beers and countries), and fuzzy datatypes. However, in this paper we discard the fuzzy information (fuzzy datatypes representing linguistic labels as possible values of some data properties, such as LowAlcohol).

The complete Fuzzy Beer has $15,317$ beer individuals. In this paper, we consider the fragment with 31 beers, 24 breweries, and 12 countries used in [20].

Let us consider the semantic query

$$Q(x) = \mathsf{Lager}(x) \wedge \mathsf{ABV}(x, 4.5) \wedge \mathsf{IBU}(x, 10) \qquad (14)$$

where ABV denotes the alcohol level and IBU the bitterness.

Let us start with OWA aggregation. The proportion of individuals for which i query terms are defined is: $N = [0.582, 0.403, 0.015]$. This means that for most individuals only the truth of one query term is known, and the least frequent case is knowing the truth of the three query terms. This leads to the vector of weights $W = [0.015, 0.403, 0.582]$, so the scores are the following ones (only beers with positive scores are shown):

⟨CoronaExtra, 0.418⟩, ⟨AlhambraReserva1925, 0.015⟩, ⟨AmbarEspecial, 0.015⟩, ⟨AmbarExport, 0.015⟩, ⟨Amstel, 0.015⟩, ⟨ AsahiSuperDry, 0.015⟩, ⟨BudLight, 0.015⟩, ⟨Budweiser, 0.015⟩, ⟨Carling, 0.015⟩, ⟨CruzcampoPremiumLager, 0.015⟩, ⟨Heineken, 0.015⟩, ⟨HijosDeRivera1906Extra, 0.015⟩, ⟨MahouCincoEstrellas, 0.015⟩, ⟨MahouClasica, 0.015⟩, ⟨MortSubiteKriekLambicTradition, 0.015⟩, ⟨PaulanerSalvator, 0.015⟩, ⟨PilsnerUrquell, 0.015⟩, ⟨QuilmesCristal, 0.015⟩, ⟨VollDammDobleMalta, 0.015⟩ .

We can see CoronaExtra is the beer with the highest score, as the system knows that it is a Lager of 4.5 ABV (its bitterness level is unknown). Then, there are 18 beers with the same score, as they both satisfy one query term, although MortSubiteKriekLambicTradition satisfies a different one, as we will see later.

Now, let us consider weighted mean aggregation. The proportion of individuals for which the truth of the query terms is known is $V = [0.940, 0.467, 0.030]$, where the positions of the arrays correspond to Lager, ABV, and IBU, in this order. We know if various individuals are a Lager (18 individuals) or not (44, including 8 beers, the breweries and the countries), some information about the alcohol (it is defined for all beers but not for the breweries), but the bitterness has more missing values (it is only defined for 2 beers). Regarding the informativeness, we have $H = [0.291, 1.265, 0.477]$. Therefore, the most informative attribute is the alcohol level, followed by the bitterness. As expected, the informativeness of these numerical data properties is much higher, as the range of possible values (real numbers) is wider.

Thus, by combining these values (V and H), the vector of weights for the weighted mean is $W = [0.313, 0.671, 0.016]$, ABV being ranked as the most important query term, followed by Lager, and IBU as the least important one. The result of the query is the following one:

⟨CoronaExtra, 0.984⟩, ⟨MortSubiteKriekLambicTradition, 0.671⟩, ⟨AlhambraReserva1925, 0.313⟩, ⟨AmbarEspecial, 0.313⟩, ⟨AmbarExport, 0.313⟩, ⟨Amstel, 0.313⟩, ⟨AsahiSuperDry, 0.313⟩, ⟨BudLight, 0.313⟩, ⟨Budweiser, 0.313⟩, ⟨Carling, 0.313⟩, ⟨CruzcampoPremiumLager, 0.313⟩, ⟨Heineken, 0.313⟩, ⟨HijosDeRivera1906Extra, 0.313⟩, ⟨MahouCincoEstrellas, 0.313⟩, ⟨MahouClasica, 0.313⟩, ⟨PaulanerSalvator, 0.313⟩, ⟨PilsnerUrquell, 0.313⟩, ⟨QuilmesCristal, 0.313⟩, ⟨VollDammDobleMalta, 0.313⟩ .

Now, CoronaExtra is again the beer with the highest score. But now, weighted mean aggregation ranks MortSubiteKriekLambicTradition in a second position, as knowing ABV is deemed more important than knowing whether it is a Lager or not. Finally, 17 Lager beers have the same score.

VII. Related work

There are some previous approaches in the literature to deal with incomplete information in Semantic Web technologies [4], but they do not rely on fuzzy logic. While fuzzy logic has inspired some query relaxation approaches (e.g., [21] uses similarity measures between individuals to solve SPARQL queries), applying aggregation operators to query relaxation has received very little attention.

One exception is [22], proposing an interactive weights refinement method for the OWA operators in the framework of query relaxation for DBMSs. However, rather than addressing missing values, it generalizes traditional conjunctive-disjunctive queries to OWA aggregation, having an intermediate behavior between a disjunctive and a conjunctive view.

Another related work is the GimmeHop beer ontology-based recommender system [20], which uses weighted mean or OWA operators to compute weight vectors of different sizes based on the number of restrictions available for each beer. OWA weights are computed from a common fuzzy quantifier using a well-known strategy called quantifier-based aggregation [23]. Weighted mean weights are computed by giving more importance to the attributes marked as more relevant by the user. In our case, instead, we compute the weights taking into account the number of missing values and the informativeness of the values.

OBSERVER was the first system to do query relaxation in SQA and quantify the loss of information [24]. As our approach, it can retrieve individuals that satisfy some of the restrictions only, but it provides a global score of the whole answer set, while our system assigns a score to each individual in the answer set, so we are able to rank the query results.

[6] http://webdiis.unizar.es/~ihvdis/GimmeHop_app

Finally, although there exist many works trying to learn the weights of OWA operators (e.g., [25]), they focus on existing data but not on missing data.

VIII. Conclusions and Future Work

In this paper we have addressed the problem of Semantic Query Answering (SQA) in scenarios with missing values. As a solution, we proposed a novel approach to SQA where the different terms of the query are combined using fuzzy aggregation operators. Therefore, individuals that satisfy some query terms but have missing values in other query terms are now retrieved as part of the answer but with a lower score.

We have discussed several ways to compute the weights of the different terms in the query in an automatic way. On the one hand, we showed how to build OWA weights from the distribution of the individuals with missing values. On the other hand, we showed that it is also possible to compute weighted mean weights from the distribution of the missing values and the informativeness of the restrictions.

We compared both approaches, showing that weighted mean is clearly preferable both for computational reasons (the most expensive part of computing the weights can be reused for different queries) and for logical properties (weighted mean is more stable to different queries, it is able to discriminate individuals with the same number of satisfied restrictions, and it does not introduce zero weights).

Our approach has been implemented in a prototype. For the sake of concrete illustration, we discussed a use case in the field of beer recommendation.

As future work, we would like to evaluate our approach using more complex datasets, ideally involving real-world data. Another promising idea is to extend our approach to query fuzzy ontologies or fuzzy knowledge graphs, where the terms of the semantic queries can be partially satisfied. We would also like to address scenarios where there are multiple ontologies rather than a single one. Finally, we would like to investigate more sophisticated strategies to estimate the informativeness of the properties, particularly in the case of multi-valued or functional properties.

Acknowledgments

The authors were partially supported by the I+D+i project PID2020-113903RB-I00, funded by MCIN/AEI/10.13039/501100011033. C. Bobed, F. Bobillo, and E. Mena were also funded by project T42_23R (funded by Gobierno de Aragón). U. Straccia was partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under (GA No 952215), by the FAIR (Future Artificial Intelligence Research) project funded by the NextGenerationEU program within the PNRR-PE-AI scheme (M4C2, investment 1.3, line on Artificial Intelligence), and by the H2020 STARWARS Project (GA No. 101086252), a type of action HORIZON TMA MSCA Staff Exchanges. Finally, we are in debt with Ignacio Huitzil for some help with the dataset.

References

[1] S. Staab and R. Studer, Eds., *Handbook on Ontologies*, ser. International Handbooks on Information Systems. Springer, 2004.

[2] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutiérrez, S. Kirrane, J. E. Labra Gayo, R. Navigli, S. Neumaier, A.-C. Ngonga Ngomo, A. Polleres, S. M. Rashid, A. Rula, L. Schmelzeisen, J. F. Sequeda, S. Staab, and A. Zimmermann, *Knowledge Graphs*, ser. Synthesis Lectures on Data, Semantics, and Knowledge 22. Morgan & Claypool, 2021.

[3] R. Reiter, "On closed world data bases," in *Logic and data bases*, H. Gallaire and J. Minker, Eds. Plenum Press, 1978, pp. 55–76.

[4] G. Fakih and P. Serrano-Alvarado, "A survey on SPARQL query relaxation under the lens of RDF reification," in *39ème Conférence sur la Gestion de Données – Principes, Technologies et Applications (BDA 2023)*, 2023.

[5] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *The Scientific American*, vol. 284, no. 5, pp. 34–43, 2001.

[6] B. Cuenca-Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler, "OWL 2: The next step for OWL," *Journal of Web Semantics*, vol. 6, no. 4, pp. 309–322, 2008.

[7] F. Baader, I. Horrocks, C. Lutz, and U. Sattler, *An Introduction to Description Logic*. Cambridge University Press, 2017.

[8] I. Horrocks, O. Kutz, and U. Sattler, "The even more irresistible $\mathcal{SROIQ}$," in *Proceedings of the 10th International Conference of Knowledge Representation and Reasoning (KR 2006)*, 2006, pp. 57–67.

[9] F. Manola and E. Miller, "RDF Primer, W3C recommendation," http://www.w3.org/TR/rdf-primer, 2004.

[10] D. Brickley and R. V. Guha, "RDF Schema 1.1, W3C recommendation," http://www.w3.org/TR/rdf-schema/, 2014.

[11] G. Beliakov, H. Bustince, and T. Calvo, *A Practical Guide to Averaging Functions*, ser. Studies in Fuzziness and Soft Computing 329. Springer, 2016.

[12] L. A. Zadeh, "Fuzzy sets," *Information Control*, vol. 8, pp. 338–353, 1965.

[13] G. J. Klir and B. Yuan, *Fuzzy sets and fuzzy logic - theory and applications*. Prentice Hall, 1995.

[14] R. R. Yager, "On ordered weighted averaging aggregation operators in multicriteria decision making," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 18, no. 1, pp. 183–190, 1988.

[15] F. Bobillo, M. Cerami, F. Esteva, À. García-Cerdaña, R. Peñaloza, and U. Straccia, "Fuzzy description logics," in *Handbook of Mathematical Fuzzy Logic Volume III*, ser. Studies in Logic, Mathematical Logic and Foundations, P. Cintula, C. Fermüller, and C. Noguera, Eds. College Publications, 2015, vol. 58, ch. XVI, pp. 1105–1181.

[16] U. Straccia, *Foundations of Fuzzy Logic and Semantic Web Languages*, ser. CRC Studies in Informatics Series. Chapman & Hall, 2013.

[17] D. Hankerson, P. D. Johnson, and G. A. Harris, *Introduction to Information Theory and Data Compression*, 1st ed. CRC Press, Inc., 1998.

[18] M. Horridge and S. Bechhofer, "The OWL API: A Java API for OWL ontologies," *Semantic Web*, vol. 2, no. 1, pp. 11–21, 2011.

[19] B. Glimm, I. Horrocks, B. Motik, G. Stoilos, and Z. Wang, "HermiT: An OWL 2 reasoner," *Journal of Automated Reasoning*, vol. 53, no. 3, p. 245–269, 2014.

[20] I. Huitzil, F. Alegre, and F. Bobillo, "GimmeHop: A recommender system for mobile devices using ontology," *Fuzzy Sets and Systems*, vol. 401, pp. 55–77, 2020.

[21] A. Hogan, M. Mellotte, G. Powell, and D. Stampouli, "Towards fuzzy query-relaxation for rdf," in *The Semantic Web: Research and Applications*. Springer, 2012, pp. 687–702.

[22] D. A. Łazar, "A fuzzy relaxation mechanism for relational dbms querying based on the ordered weighted averaging (OWA) operators," *Control and Cybernetics*, vol. 25, no. 2, pp. 381–402, 1996.

[23] R. R. Yager, "Quantifier guided aggregation using OWA operators," *International Journal of Intelligent Systems*, vol. 11, no. 1, pp. 49–73, 1996.

[24] E. Mena, A. Illarramendi, V. Kashyap, and A. P. Sheth, "OBSERVER: an approach for query processing in global information systems based on interoperation across pre-existing ontologies," *Distributed Parallel Databases*, vol. 8, no. 2, pp. 223–271, 2000.

[25] A. Cristóbal, I. Huitzil, and F. Bobillo, "Learning OWA weights by combining fuzzy quantifiers with empirical data," in *Actas del XX Congreso Español sobre Tecnologías y Lógica Fuzzy (ESTYLF 20–21)*, 2021, pp. 351–356.