Enhancing Soft Web Intelligence with User-Defined Fuzzy Aggregators

Paolo Fosci[a] and Giuseppe Psaila[b]
University of Bergamo - DIGIP, Viale Marconi 5, 24044 Dalmine (BG), Italy
paolo.fosci@unibg.it, giuseppe.psaila@unibg.it

Keywords: Soft Web Intelligence, User-Defined Fuzzy Aggregators, Novel Constructs in J-CO-QL+, Soft Querying on JSON Data Sets.

Abstract: In our previous work, we proposed Soft Web Intelligence as the interpretation of the general notion of Web Intelligence in the current technological panorama, in such a way *JSON* data sets are acquired from the Internet, stored within *JSON* document stores and then processed and queried by means of soft computing and soft querying methods. Specific extensions to the *J-CO* Framework and to its query language (named *J-CO-QL*$^+$) made possible to practically implement the concept.

However, any "data intelligence" activity does not exclude aggregating data, but *J-CO-QL*$^+$ did not provide statements for defining "user-defined fuzzy aggregators". In this paper, we present the novel constructs introduced into *J-CO-QL*$^+$ to allow users to define and use their own fuzzy aggregators, so as to evaluate membership degrees to fuzzy sets moving from array fields within processed *JSON* documents. This way, complex soft queries are enabled, so as to enhance Soft Web Intelligence.

1 INTRODUCTION

Two decades ago, Web Intelligence was proposed in (Yao et al., 2001) as an approach to exploit the large amount of data and information that is possible to obtain from the World-Wide Web; the foreseen key technology was Artificial Intelligence (AI), because it was recognized that the semi-structured or totally unstructured form of information that is published on the Web made classical approaches to Business Intelligence substantially unsuitable.

Two decades later, the format that has become very popular is *JSON* (JavaScript Object Notation), due to its syntactic simplicity and its ease to be processed in programming languages. This success has been accompanied by the advent of *JSON* document stores, i.e., DBMSs (Database Management Systems) that natively store and query *JSON* data sets. Consequently, data scientists and data engineers very often deal with Web Intelligence scopes in which they have to gather, integrate, query and publish *JSON* data sets.

In (Fosci and Psaila, 2022b), we envisioned the notion of "Soft Web Intelligence": soft computing and soft querying (based on Fuzzy-Set Theory) can actually provide the tools to perform Web Intelligence tasks that must process *JSON* data sets coming from web sources. Fuzzy-Set Theory and Fuzzy Logic are indeed tools that belong to the AI world, consequently we conceived the idea of "Soft Web Intelligence" as a natural evolution of Web Intelligence.

The *J-CO* Framework is a pool of software tools that is under development at the University of Bergamo (Italy); its goal is to provide analysts and data engineers with sophisticated capabilities to gather, integrate and query *JSON* data sets; its query language, named *J-CO-QL*$^+$, is undergoing a continued evolution with the addition of novel constructs: specifically, we are currently introducing constructs for further extending its capabilities for performing soft querying on *JSON* data sets. In order to achieve a practical support to our vision, in (Fosci and Psaila, 2022b) we introduced extensions to the *J-CO* Framework specifically designed to practically realize *Soft Web Intelligence*; furthermore, through a practical case study, we showed that they are effective.

A typical computational activity that is performed in Business Intelligence is "aggregating measures of facts", so as to provide an aggregated view of events described by the analyzed data. It is reasonable to guess that aggregating data should be a typical task to do in *Soft Web Intelligence*, but in our previous works on *J-CO-QL*$^+$ aggregation was not considered, because times were not mature. Now, it is the time to introduce aggregators in *J-CO-QL*$^+$, to further extend

[a] https://orcid.org/0000-0001-9050-7873
[b] https://orcid.org/0000-0002-9228-560X

Fosci, P. and Psaila, G.
Enhancing Soft Web Intelligence with User-Defined Fuzzy Aggregators.
In *Proceedings of the 19th International Conference on Web Information Systems and Technologies (WEBIST 2023)*, pages 258-267
ISBN: 978-989-758-672-9; ISSN: 2184-3252

the support to *Soft Web Intelligence*; specifically, we consider "fuzzy aggregators", which can be used to rank documents by aggregating either values in array fields or membership degrees to multiple fuzzy sets.

In this paper, we propose a novel construct, named `CREATE FUZZY AGGREGATOR`, that we added to *J-CO-QL*$^+$ so as to define "user-defined fuzzy aggregators"; its clauses drive users through the definition of a fuzzy aggregator, giving them the intuition of the semantic model. Through a practical case study, we will show how to exploit fuzzy aggregators in a scope of *Soft Web Intelligence*.

The paper is organized as follows. Section 2 presents the background of our work. Section 3 introduces the vision of *Soft Web Intelligence* and presents the main features of the *J-CO* Framework. Section 4 introduces the novel construct that we added to *J-CO-QL*$^+$ to declare user-defined fuzzy aggregators, together with the semantic model. Section 5 addresses a practical case study by exploiting user-defined fuzzy aggregators. Finally, Section 6 draws conclusions and future work.

2 RELATED WORK

This paper connects two areas that usually are not considered together, i.e., Web Intelligence and fuzzy logic. This is not the first attempt we make, but now we consider fuzzy aggregators (that were not considered in our previous work (Fosci and Psaila, 2022b)).

Specifically, Web Intelligence was introduced in (Yao et al., 2001) to obtain useful information from Web content. The complexity of Web content suggested to rely on Artificial Intelligence (AI) to get rid of this complexity. But what is AI? Techniques for "Data Mining" are certainly considered as AI techniques and are exploited in Web Intelligence (Han and Chang, 2002), as well as neural networks are nowadays perceived as "the AI".

Nonetheless, in the literature, many papers tries to give a more specific yet wider interpretation of Web Intelligence, such as "Computational Web Intelligence" (Zhang and Lin, 2002), i.e., the adoption of "Computational Intelligence" in Web Intelligence, as well as "Brain Informatics" (Zhong et al., 2006), i.e., fostering Web Intelligence through techniques that come out from the study of the human brain.

Fuzzy Logic and Soft Computing belong to AI too: indeed, its capability of approximate reasoning based on "linguistic predicates" provides a significant contribution towards AI. Consequently, it is straightforward that fuzzy logic can be exploited for Web Intelligence. Zadeh, the creator of Fuzzy-Set Theory (Zadeh, 1965), had this vision clearly in his mind: indeed, in (Zadeh, 2004a; Zadeh, 2004b), he showed that soft computing could play an important role.
Remember that a Fuzzy Set A in a universe U is a mapping for each $x \in U$, $A : x \rightarrow [0,1]$, also denoted as $\mu_A : x \rightarrow [0,1]$. The co-domain is the set of "membership degrees" (or simply "memberships", for brevity in the following): each item x belongs to A with a degree; when the degree is $0 < \mu_A(x) < 1$, x belongs to A only partially; obviously, $\mu_A(x) = 1$ denotes full membership of x to A, while $\mu_A(x) = 0$ means hat x does not belong at all to A.

However, looking for papers about fuzzy logic and soft computing in Web Intelligence, very few works can be found. The paper (Kacprzyk and Zadrożny, 2010) exploits soft computing in a group decision-making system to express preferences, but Web Intelligence activities were not supported by soft computing. The paper (Poli, 2015) uses `FUZZYALGOL`, a fuzzy procedural programming language (Reddy, 2010), for soft querying Web sources.

Consequently, this is why in (Fosci and Psaila, 2022b) we proposed the concept of *Soft Web Intelligence*, trying to give a modern interpretation of the concept of Web Intelligence on the basis of the current technological panorama; in Section 3.1, we define the concept in a precise way.

Nevertheless, Data Intelligence activities (Alahakoon and Yu, 2015) ask for aggregation; consequently, *Soft Web Intelligence* asks for "fuzzy aggregations" (since it relies on fuzzy sets). A plethora of proposals for fuzzy aggregators can be found in the literature. Many of them, such as "t-norm" and "t-conorm" operators, see (Farahbod and Eftekhari, 2012), consider the aggregation of "pairs of items", for example the classical `AND` and `OR` operators in the fuzzy version. In this paper, we are focused on groups (or categories) $G_j = \{x_{j,1}, x_{j,2}, \dots\}$) of items $x_{j,i}$ that belong to the same group G_j because they share some common properties or are samples of the same category of items. Each $x_{j,i}$ singularly may belong to a fuzzy set A, thus it is provided with a membership $\mu_A(x_{j,i})$. Consequently, the set $\overline{A}$ of groups G_j can be seen as a partition of A: with this vision, the membership of a group G_j to $\overline{A}$ should be derived by somehow aggregating memberships $\mu_A(x_{j,i})$, group by group. Alternatively, if $x_{j,i}$ has not a membership, its values might have to be aggregated to obtain the final membership of the G_j group.
Popular fuzzy aggregators of this type are "Weighted aggregation" (see (Dombi and Jónás, 2022)) and "Ordered Weighted Aggregation" (OWA) (Yager, 1988; Li and Yen, 1995).

3 SOFT WEB INTELLIGENCE AND THE J-CO FRAMEWORK

In this section, we introduce our vision about *Soft Web Intelligence*, moving from the original paper (Fosci and Psaila, 2022b). Then, we briefly introduce the *J-CO* Framework, which is the technical tool that has inspired the idea of *Soft Web Intelligence*.

3.1 Soft Web Intelligence

We conceived the notion of *Soft Web Intelligence* as an evolution of the generic idea of Web Intelligence, on the basis of the current technological panorama. The following definition provides a synthetic characterization of the concept, that has come out while working on it after (Fosci and Psaila, 2022b).

Definition 1. *"Soft Web Intelligence" is the continued acquisition, integration and querying of JSON data sets coming from or representing Web sources, by exploiting Soft Computing and Soft Querying, so as to use them for decision making and knowledge discovery.*

In some sense, we could say that Definition 1 instantiates the very generic definition of *Web Intelligence* provided more than 20 years ago (Yao et al., 2001). It is the result of the considerations made by Zadeh in (Zadeh, 2004a; Zadeh, 2004b), concerning the fact that *JSON* has imposed, in place of XML, as the most popular format to represent and share data over the Internet, as well as it is the result of the availability of *NoSQL* DBMSs known as "*JSON* document stores", which natively store *JSON* data sets.

Through Figure 1, we illustrate how we figure out *Soft Web Intelligence*.

- **Web Sources**. Different kinds of Web sources can be considered. Users usually think about Web pages, but many services provide *JSON* data sets. For example, Web Services can be contacted to provide either complete data sets or single pieces of data; this latter ones could be singularly collected into a global data set; typically, social media expose Web services that can be exploited to interact with the system; nowadays, *JSON* is the data format on which most of Web services rely.
  Open-Data portals are a common channel that is exploited by Public Administrations to publish data sets (possibly "Authoritative Data Sets") concerned with the administered territory or country. Among all formats, *JSON* and *GeoJSON* are becoming more and more popular in this context too. As a final example, the content of Web pages (i.e., HTML pages) could provide useful data sets and information. In this case, techniques of "Web Scraping" could be exploited for acquiring the content of HTML pages and represent their information content as *JSON* data sets.

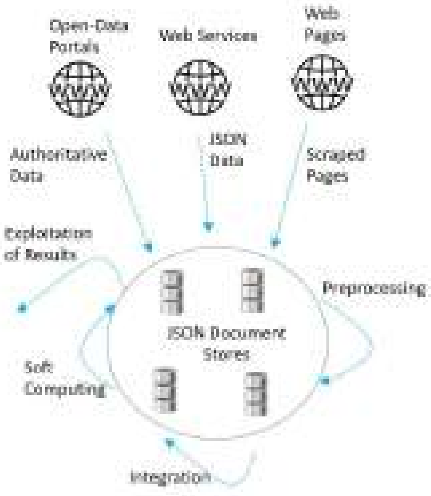

Figure 1: Vision of Web Intelligence.

- ***JSON* Document Stores**. A pool of databases managed by (potentially different) *JSON* document stores is the right solution to store *JSON* data sets acquired from Web sources. A very famous *JSON* store is *MongoDB*, but other products are available (such as *CouchDB*, chosen as the DBMS for the *HyperLedger Fabric* permissioned BlockChain platform (Bringas et al., 2019)).
- **Processing Activities**. Processing is the key factor for the success of *Soft Web Intelligence*. Certainly, "Preprocessing" is the first activity to perform on data, so as to remove noise and make formats homogeneous. The second activity is "Integration", i.e., integrating the different data sets possibly stored in different databases, by uniting those pieces of information that came from different sources. The final activity is "Soft Computing", in which techniques based on soft computing and soft querying are used to extract knowledge from integrated data. Clearly, the more powerful the support for soft computing and soft querying, t he greater the possibility to extract useful data and knowledge from the acquired data sets.

3.2 The J-CO Framework

The *J-CO* Framework is a pool of software tools whose goal is to provide analysts with a powerful support for gathering, integrating and querying possibly-large collections of *JSON* data sets. The core of the framework is its query language, named *J-CO-QL*$^{+}$.

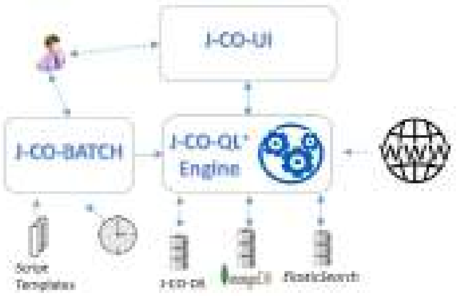

Figure 2: The *J-CO* Framework.

The current organization of the framework, which is the result of (Fosci and Psaila, 2022b), is depicted in Figure 2. We explain it hereafter.

- *J-CO-QL$^+$ Engine*. This component actually executes *J-CO-QL$^+$* scripts (i.e., queries). It is able to retrieve data from document databases (for example, managed by *MongoDB*) and save results into them; it also can send HTTP requests to Web sources to get *JSON* data sets directly from them.
- *J-CO-DS*. This component is a simple document store specifically introduced in (Psaila and Fosci, 2018) to store large or very large single *JSON* documents (such as many *GeoJSON* documents that cannot be stored within other *JSON* stores, such as *MongoDB*). In (Fosci and Psaila, 2022b), we extended its data model, in such a way it now supports three different types of collections:
  - *Static Collections* are the classical collections that contains *JSON* documents, similarly to other *JSON* stores (the content of a collection can be updated by the user or by the *J-CO-QL$^+$ Engine*);
  - *Dynamic Collections* automatically acquire the content of web sources at scheduled times, so as to provide images of web sources without need to access the Internet when they are processed;
  - *Virtual Collections* are associated to Web sources, but they do not manage any local copy of them, in that Web sources are accessed when the virtual collection is accessed by users or the *J-CO-QL$^+$ Engine* (thus, virtual collections provide a database view of Web sources).
- *J-CO-BATCH* (Introduced in (Fosci and Psaila, 2022b)) is an off-line executor of *J-CO-QL$^+$* scripts; in particular, it is possible to schedule the repeated execution of scripts. Another feature is the concept of "template": *J-CO-QL$^+$* scripts can be parameterized, so as to reuse them with different settings/configurations.
- *J-CO-UI* is the user interface, by means of which analysts can write *J-CO-QL$^+$* scripts, submit them to the *J-CO-QL$^+$ Engine* and inspect results.

```
CREATE FUZZY AGGREGATOR integrateRain
  PARAMETERS   rainData TYPE ARRAY
  FOR ALL      rd IN rainData
    AGGREGATE  rd AS av
  EVALUATE     av
  POLYLINE [ (  0, 0.0),   ( 50, 0.0),   (100, 0.1),
             (200, 0.7),   (300, 0.9),   (400, 1.0) ];
```

Listing 1: *J-CO-QL$^+$*: fuzzy aggregator `integrateRain`.

Data and Execution Models. For the sake of clarity, we briefly introduce the data and execution models of *J-CO-QL$^+$*.

- A *JSON* document is the basic computational unit. *JSON* documents are grouped within "collections": an instruction takes one or two collections as input and generates a new collection.
- A query or script in *J-CO-QL$^+$* is a sequence of instructions. They constitute a "piped flow", in such a way an instruction receives a "temporary collection" (generated by the previous instruction) as input and possibly generates a novel temporary collection as output. The "temporary" adjective denotes that the collection is not saved in any database, but is a temporary result of the process. Instructions can also acquire data either from *JSON* databases or from Web sources.
- For each single document, it is possible to independently evaluate its memberships to many fuzzy sets. These degrees are represented within the same document, by adding a special root-level field named `~fuzzysets`. It is a nested document that behaves as a key/value map: a field within it denotes the membership to a fuzzy set, in such a way the field name is the name of the fuzzy set, while the value (in the range $[0,1]$) is the degree. Specific clauses of *J-CO-QL$^+$* instructions evaluate soft conditions, by evaluating memberships.

4 USER-DEFINED FUZZY AGGREGATORS

User-defined fuzzy aggregators were missing in *J-CO-QL$^+$*: indeed, when documents in collections contain arrays, it is highly probable that their content should be somehow summarized, so as to determine the membership to some fuzzy set and contribute to soft querying. Hereafter, we present the novel statement named `CREATE FUZZY AGGREGATOR`.

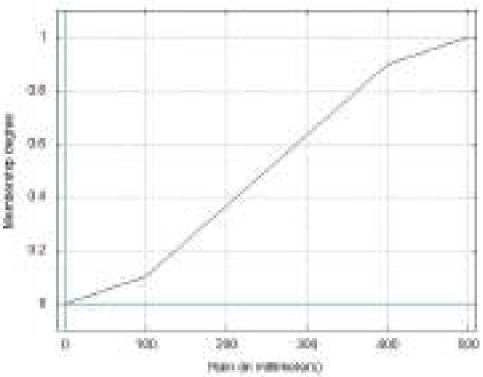

Figure 3: Membership function for the fuzzy aggregator `integrateRain`.

4.1 Basic Structure

Listing 1 reports the definition of a very simple fuzzy aggregator: its goal is to perform a cumulative aggregation (sum) of numerical values within an array. Hereafter, we explain it in details.

- The fuzzy aggregator is named `integrateRain`, as specified after the keywords `CREATE FUZZY AGGREGATOR`. The name is motivated by the fact that the aggregator has to aggregate the amount of rain (in millimeters).
- The clause `PARAMETERS` defines the formal parameters for the aggregator: in this case, only one parameter, named `rainData` is specified, which must be an array. Specifically, the aggregator is expected to receive the amount of rain, so each item in the array `rainData` denotes millimeters of rain.
- The clause `FOR ALL` scans all values in an array parameter, so as to perform aggregations. Specifically, all items `rd` in the array `rainData` are explored.
- The sub-clause `AGGREGATE` evaluates, for each item `rd` in the array `rainData`, an expression, whose value is aggregated into the "variable" specified after the keyword `AS`.
  Specifically, the expression to evaluate refers only to the item `rd`; this means that their values are aggregated into the value `av`.
- The clause `EVALUATE` is evaluated after the clause `FOR ALL` has scanned all items in the array and generated an aggregated value.
  Specifically, it refers to the `av` aggregated value only, meaning that the aggregated value is taken as it is. Clearly, since we aggregated generic numerical values, the result is not in the range $[0,1]$, so it cannot be considered as a membership. The next clause converts it into a membership.
- The clause `POLYLINE` defines a membership function as a polyline of points: while the x coordinate can be any real number, the y coordinate must be necessarily in the range $[0,1]$, because the membership function converts the value returned by the clause `EVALUATE` into a membership value.
  The polyline is depicted in Figure 3: notice that we considered as range of interest from 0 *mm* to 500 *mm* of rain (which is really a lot of rain). With 500 *mm*, the value 1 for the membership is reached: greater values of rain still obtain 1 as membership.

Resuming, the `integrateRain` fuzzy aggregator moves from an array of values, aggregates them and generates a membership to a fuzzy set.

```
CREATE FUZZY AGGREGATOR owaRain
  PARAMETERS    rainData TYPE ARRAY
  SORT          rd IN rainData
    BY          rd TYPE NUMERIC ASC AS sRainData
  FOR ALL       srd IN sRainData
    LOCALLY     ( POS^2 - (POS-1)^2 )
                  / (COUNT(sRainData)^2) AS w
    AGGREGATE   srd * w AS av
  EVALUATE      av
  POLYLINE      [(0.00, 0.0), (0.10, 0.0), (0.15, 0.7),
                 (0.20, 0.8), (0.50, 0.9), (0.80, 1.0)];
```

Listing 2: *J-CO-QL*$^{+}$: fuzzy aggregator `owaRain`.

4.2 Adding Local Derived Values

Listing 2 shows a more complex fuzzy aggregator, named `owaRain`. It performs the "Ordered Weighted Aggregation" (OWA, see (Yager, 1988)): a monotone function is used to determine the weights of each item in the array to aggregate, in such a way the array is previously sorted. Hereafter, we explain the aggregator reported in Listing 2 in details.

- As for the aggregator reported in Listing 1, the aggregator `owaRain` receives one single array parameter, named `rainData`.
- The clause `SORT` generates a novel array, named `sRainData`, sorting each numeric item `rd` in the array `rainData` in ascending order.
- The clause `LOCALLY` evaluates, for each item `srd` in the array `sRainData`, a derived value that is used in the sub-clause `AGGREGATE`.
  Specifically, the $f(x) = x^2$ is used to compute the weight `w` of the item `srd`, whose position in the array `sRainData` is denoted by `POS` (the first item has `POS` $= 1$). Formally, the weight of the i-th item in `sRainData` is
  `w`$= f(i/|$`sRainData`$|) - f((i-1)/|$`sRainData`$|)$.
  Since f is a parable, items with highest position get the highest weights; furthermore, the array is