Nested Forms with Dynamic Suggestions for Quality RDF Authoring

Pierre Maillot, Sébastien Ferré, Peggy Cellier, Mireille Ducassé, Franck Partouche

Nested Forms with Dynamic Suggestions for Quality RDF Authoring

Pierre Maillot[1], Sébastien Ferré[1], Peggy Cellier[2], Mireille Ducassé[2], and Franck Partouche[3]

[1] IRISA/Université de Rennes 1[**]
[2] IRISA/INSA Rennes
Campus de Beaulieu, 35042 Rennes cedex, France
{firstname}.{lastname}@irisa.fr
[3] IRCGN, 5 Boulevard Hautil, 95000 Cergy, France
franck.partouche@gendarmerie.interieur.gouv.fr

Abstract. Knowledge acquisition is a central issue of the Semantic Web. Knowledge cannot always be automatically extracted from existing data, thus domain experts are required to manually produce it. On the one hand, learning formal languages such as RDF represents an important obstacle to non-IT experts. On the other hand, well-known data input interfaces, such as forms, do not address well the relational nature and flexibility of RDF. Furthermore, it is difficult to maintain data quality through time, and across contributors. We propose FORMULIS, a form-based interface for guided RDF authoring. It completely hides RDF notations, addresses the relational aspects with nested forms, and guides users by computing intelligent filling suggestions. Two user experiments show that FORMULIS helps users maintain good data quality, and can be used by users without Semantic Web knowledge.

Keywords: Knowledge acquisition, RDF graph, Forms, Dynamic suggestions, Query relaxation, Semantic Web

1 Introduction

The development of the Semantic Web [2] is fueled by the constant creation of data. RDF graphs are created either through automatic extraction from existing sources or from manual acquisition by contributors. The latter is required when there is no digital source (e.g., description of forged ID documents) or when the extraction cannot easily be automated (e.g., image understanding). There are a number of issues with RDF authoring by domain experts. A first issue is that contributors have to learn the formal syntax and semantics of RDF. This learning effort is an obstacle to the adoption and growth of the Semantic Web. A second issue is that the graph structure of RDF is difficult to present in the user interface to contributors, who are mostly used to forms and tables. A third

[**] This work is supported by ANR project IDFRAud (ANR-14-CE28-0012)

issue is that RDF is very flexible, i.e. every entity can be related to any other entity through any property. A too flexible input interface will accept data that is inconsistent with the intended schema, while a schema-restricted input interface will forbid contributors to extend or amend the schema when needed.

A number of RDF authoring tools based on forms have been proposed, e.g. WebProtégé [18] or ActiveRaUL [6]. They offer a user-friendly interface but require a Semantic Web expert to configure the forms before domain experts can use them. It works for datasets with a stable ontology but not for datasets that are new or that continually evolve. Another limitation is that they do not allow to create graphs composed of several linked entities, working on several entities at the same time. In addition, those forms guide user input in the knowledge base thanks to simple consistency rules (e.g., film actors must be persons). However, in general, those rules are too simple and do not sufficiently reduce the suggestions. It forces the user to scroll through long lists of potentially irrelevant values (e.g., all persons when looking for an actor). That makes data authoring a tedious and error-prone process. On the opposite, SEWELIS [10] allows domain experts to design the ontology bottom-up, while filling up the base. It is very flexible, and it has a dynamic suggestion mechanism that helps domain experts input consistent data. However, its interface requires too many micro-actions to build entity descriptions.

We propose FORMULIS, a system that offers the user-friendliness of forms while producing arbitrary RDF graphs, and guiding users with dynamic suggestions based on existing data. During the creation of an entity by a domain expert, the principle is to retrieve similar entities based on already filled fields, to show them, and to make suggestions from their fields and values. Those similar entities are recomputed after each additional input in the form. The contribution to form-based RDF authoring is threefold. First, the forms can be nested as deeply as necessary so as to create several interlinked entities at once. Second, FORMULIS dynamically suggests fields and values that take into account all the fields and values already entered in the base and in the current form, thanks to SEWELIS query relaxation mechanisms. Last but not least, the forms can be extended with new fields and new sub-forms at any time, according to user needs and data evolution.New forms can also be created on the fly for new classes of resources. This enables to leverage the flexibility of the RDF data model, and removes the need for the configuration of the interface by Semantic Web experts. Two user experiments show that FORMULIS helps users maintain good data quality, and can be used by users without Semantic Web knowledge. The first one is a controlled experiment involving laymen on the description of cooking recipes. The second is an application in a real setting involving forensic experts from IRCGN (Forensic Science Institute of the French Gendarmerie) on the description of forged Portuguese ID cards seized during a police operation.

Section 2 briefly recalls the principles of SEWELIS needed for the understanding of the remainder. Section 3 describes FORMULIS. Section 4 describes the experiments and their results. Section 5 discusses related work and Section 6 concludes the article.

2 Preliminaries

In this section, we briefly present SEWELIS[4], a Semantic Web system to query and update RDF graphs, used in the work presented in this paper. It has been implemented in a larger framework called Logical Information Systems (LIS). LIS emerged from the need to reconcile the expressivity of formal languages like SPARQL [17], and the usability of navigational and interactive user interfaces such as Faceted Search (FS) [15]. The reconciliation relies on the theory of Query-based Faceted Search (QFS) [8]. The general principle is to guide users in the incremental construction of complex sentences in a formal language, e.g. SPARQL queries or RDF descriptions. Guidance makes it unnecessary for users to master the formal syntax, they only have to read sentences, not to write them. An immediate advantage is the total avoidance of syntax errors. A key aspect of LIS is that guidance is not only based on syntax but also on data in order to avoid some semantic errors as well. An example of semantic error is a query that returns no result either because it does not respect the data schema, or simply because some information is missing. This is similar to faceted search where, after selecting some criteria, only compatible and relevant criteria are suggested to refine the current selection. SEWELIS exists both as a desktop application, and as an HTTP/XML server. It can therefore be used similarly to SPARQL endpoints. The motivation for using SEWELIS rather than more established SPARQL engines and servers (e.g., Virtuoso) is that SEWELIS offers query relaxation mechanisms [10]. The latter plays an important role in the suggestion capabilities of FORMULIS.

3 Proposed approach: FORMULIS

FORMULIS aims at facilitating RDF authoring without knowledge of RDF by proposing a familiar interface, forms, and by dynamically suggesting values from current and previous inputs. On the one hand, graph-based data representations tend to become quickly illegible with the size of the graph, and text-based notations such as Turtle need prior knowledge to be understood. Forms, on the other hand, are a common interface for data input. FORMULIS generates a data-driven form-based interface for the production of RDF data. In the following, we first explain the interaction loop before describing its different parts.

3.1 The Interaction Loop

Figure 1 summarizes the interaction loop of FORMULIS, which is detailed in the following subsections. The form is initially empty, and is progressively filled through user actions (field selection, value selection or input, . . .), until it is judged complete by the user. At each loop, the partially-filled form is translated into an initial query that is supposed to retrieve the possible values for the field

[4] http://www.irisa.fr/LIS/softwares/sewelis

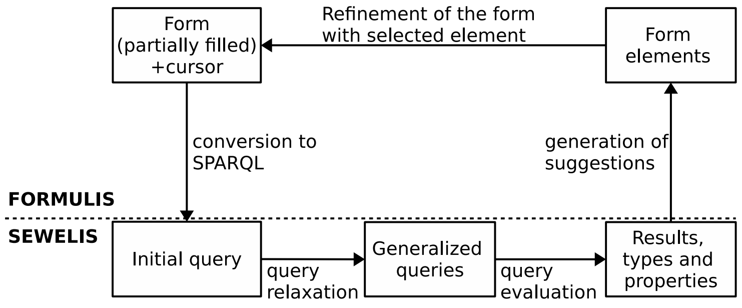

Fig. 1. Interaction loop of FORMULIS.

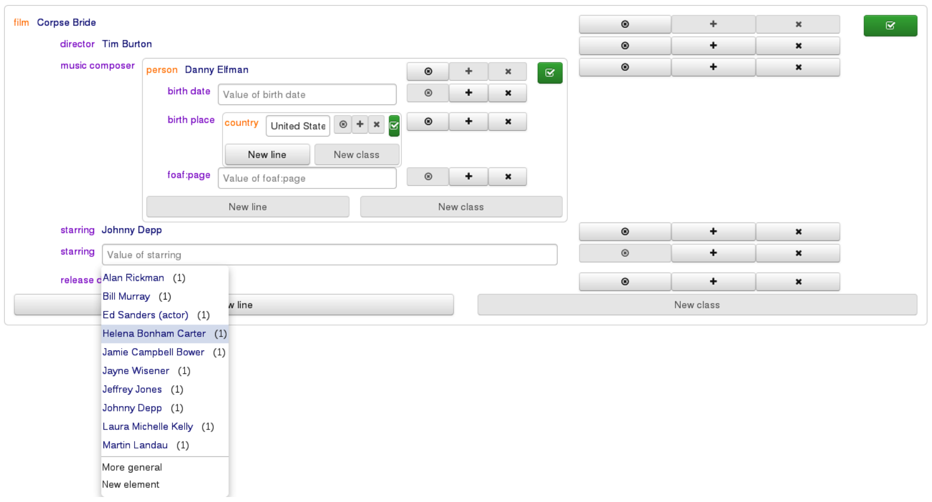
film Corpse Bride
director Tim Burton
music composer person Danny Elfman
birth date Value of birth date
birth place country United State
foaf:page Value of foaf:page
starring Johnny Depp
starring Value of starring
release d

Fig. 2. Screenshot of FORMULIS during the creation of the "Corpse Bride" film, showing the creation of a new person in a nested form as object of the property "music composer" and creating a new country as object of property "birth place".

under focus, with respect to the already filled fields. From this query, suggestions are computed with the help of the query relaxation and query evaluation mechanisms of SEWELIS. A relaxation of that initial query is in general necessary to ensure the existence of results. In order to propose legible suggestions, those results have then to be rendered as form elements, i.e. user interface widgets and controls. Finally, each time a user selects an element or activates a control, the form is modified by filling a field, adding a field, inserting a nested form, etc. From there, a new interaction step can start.

3.2 Nested Forms and their RDF Counterpart

A form F in FORMULIS is composed of an RDF class C, a resource URI R, and a set of RDF properties $p_1, \dots, p_n$, along with their respective sets of values $V_1, \dots, V_n$. Class C (e.g., `dbo:Film`) determines the type of the resource being described, and resource R (e.g., `dbr:Corpse_Bride`) determines its identity. Each property p_i (e.g., `dbo:director`) defines a field of the form. Each field may have one or several values. Indeed, unlike tabular data, the RDF data model allows a resource to have several values for the same property, and it is therefore important to account for this in an RDF editing tool. A good example about films is the property `dbo:starring` relating films to actors. A field value can be one of: a literal (e.g., a string or a date); void (i.e. the field has not yet been filled); a RDF resource already present in the knowledge base (e.g., `dbr:Tim_Burton`); or a nested form to create and describe a new RDF resource. Those definitions exhibit the recursive nature of forms in FORMULIS, which allows for nested forms. They allow for cascading descriptions of resources that transcribe the graph structure of the RDF base to the user.

3.3 User Interface

Figure 2 shows an example of nested forms, as it is displayed in the user interface of FORMULIS. The display follows the classical layout of forms with each field on a row. Each field is composed of the label of the property, and an input widget. When a property has several values (e.g., "starring"), the field is repeated. The labels of the class and the resource of the form are found at the top, above the first field. The main form is about film "Corpse Bride". The "director" field is filled with "Tim Burton", the "release date" field is filled with year 2005, and the first "starring" field is filled with "Johnny Depp". Note that RDFS labels are shown instead of URIs. The value of the "music composer" field is a nested form in order to create a new person. To illustrate the arbitrary nesting of forms, a deeper nested form has been opened to create a new country as the birth place of the new person being created as the music composer.

In order to actually produce new RDF data, the algorithm needs to translate a form into RDF when it is deemed complete by the user. The translation to Turtle is relatively straightforward. As a preprocessing step, the algorithm removes all void values, and it removes fields with no values. Then for each form F with class C, resource R, and properties $p_1, \dots, p_n$ with their associated sets of values $V_1, \dots, V_n$, it produces the turtle description: R `a` C`;` `...;` p_i $v_{i,1}$`,` `...,` v_{i,k_i}`;` `...` `.` This must be applied to the main form as well as to each nested form. For the forms in Figure 2, the algorithm obtains three Turtle descriptions (using DBpedia vocabulary), which are then sent to SEWELIS for insertion in the knowledge base.

```
dbr:Corpse_Bride a dbo:Film; dbo:director dbr:Tim_Burton;
    dbo:releaseDate "2005"^^xsd:gYear; dbo:starring dbr:Johnny_Depp.
    dbo:musicComposer dbr:Danny_Elfman;
dbr:Danny_Elfman a dbo:Person; dbo:birthPlace dbr:United_States.
dbr:United_States a dbo:Country.
```

3.4 Suggestions

In order to suggest values for a field, we need to translate the form to a SPARQL query that retrieves existing values for that field in the knowledge base. Let us suppose that the user has set the focus on the second "starring" field in Figure 2 because s-he wants to get suggestions about the other actors of the film. A simple solution would be to retrieve all values of the "starring" property in the base, using query: `SELECT ?v WHERE {?x dbo:starring ?v}`. However, this would make suggestions unspecific, here all actors present in the base would be listed. We make suggestions dynamic by taking into account the fields that have already been filled. The principle is to generate a SPARQL query by using a variable for each created resource, for each field value that is not void, and for the void value under focus. The form contents is translated into triple patterns and equality filters. The projected variable is the focus variable. In the above example, we obtain the following query, which is sent to SEWELIS for evaluation, and also for relaxation if it has empty results.

```
SELECT ?f WHERE {
    ?a a dbo:Film; dbo:director ?b ; dbo:releaseDate ?c ;
    dbo:musicComposer ?d ; dbo:starring ?e, ?f .
    ?d a dbo:Person; dbo:birthPlace ?g . ?g a dbo:Country .
    FILTER (?b = dbr:Tim_Burton)
    FILTER (?c = "2005"^^xsd:gYear)
    FILTER (?e = dbr:Johnny_Depp) }
```

3.5 Query Relaxation and Evaluation

The query received by SEWELIS may have empty results. In the above example, this is the case if the film being described is the first in the base to be directed by Tim Burton. The more fields are filled, the more specific the query is, and the more likely it is to have empty results. The second and essential step is then to relax that initial query in order to have generalized queries, and hence more query results. SEWELIS applies relaxation rules, inspired by Hurtado et al. [11], that can replace a class by a super-class, a property by a super-property, or remove altogether a triple or an equality. The *relaxation distance* of a generalized query is the number of relaxation rules that have to be applied to generate it. For example, the generalized query `SELECT ?f WHERE { ?a a dbo:CreativeWork ; dbo:director ?b ; dbo:starring ?f. FILTER (?b = dbr:Tim_Burton) }` is at relaxation distance 9: one rule for replacing `dbo:Film` by the super-class `dbo:CreativeWork`, and 8 times another rule for removing triple patterns and equalities. SEWELIS is equipped with an algorithm [10] that generates the results of generalized queries, by increasing relaxation distance. It is efficient by using dynamic programming principles to improve scalability by avoiding the enumeration of generalized queries. The query results are a set of RDF resources or literals. In the example, it is a set of actor URIs. That set can be made larger by increasing the relaxation distance.

SEWELIS does not only return a set of resources and literals, but also the types and properties that they have in the knowledge base. In the example, it would not only return a set of actors (e.g., resource `dbr:Johnny_Depp`), but also their types (e.g., class `dbo:Person`), and the properties that apply to them (e.g., properties `dbo:birthDate` and `dbo:birthPlace`). Those are then used by FORMULIS to provide suggestions to help users further to fill the form.

3.6 Generation of Suggestions and Refinement of the Form

The refinement of a form in FORMULIS is done through interactions with the form elements. Before entering into the interaction loop of a form, users are presented with a list of classes to select the type of the instance to be created. This list is generated from the suggestions returned by SEWELIS for the query `SELECT ?c WHERE { ?c a rdfs:Class }`. Once the user has selected a class, the fields of the form are initialized by the properties obtained from SEWELIS, *e.g.* a film creation form would use the query `SELECT ?p WHERE { ?s a dbo:Film. ?s ?p ?o. }`. In the situation where there is only one proposed class, the form is automatically initialized from it. After that, users enter the interaction loop to create a new resource.

Continuing with the example in previous sections, FORMULIS receives a set of results, here actors, along with their types and properties, because the user has set the focus on the second "starring" field. Suggestions in FORMULIS are derived from those results. Each result, a resource or a literal, is suggested as a value to fill the field. From these suggestions, the user can select an existing value in the displayed drop-down menu as in Figure 2. The suggested values shown in the drop-down menu are sorted by their number of occurrences in similar films as computed by SEWELIS. Users can also get the list of all values used for the selected field by requesting SEWELIS to apply maximal query relaxation.

The creation of new values is also guided by suggestions. When users choose to create a new value, FORMULIS scans the suggested values to determine the most likely class or datatype of the new value. If all suggestions are literals, the user is given a creation widget enabling the creation of dates, numbers, text or a new resource. The creation widget is then set by default on the datatype appearing most often in the suggestions. If all suggestions are resources, the property field is replaced by a new FORMULIS form. This form is nested in the current form, *e.g.* in Figure 2 a new person is created as music composer and for this new person, a new country is created. This new form is generated in the same way as the root form with the exception that its property or value suggestions are generated with queries including the filled fields of all other forms, either root or nested.

FORMULIS also enables the modification of the form independently of suggestions. Users can respectively clear filled fields, duplicate, and delete fields with the three buttons at the end of each line, as in Figure 2. It is also possible to extend the data schema by creating new fields containing new properties with the "new line" button appearing at the bottom of each form.

4 User Experiments

We have conducted two user experiments on the collaborative creation of a knowledge base. The first one compares FORMULIS with WebProtégé, a collaborative ontology editor, in a controlled experiment with layman users describing cooking recipes. The second one shows the benefits of FORMULIS to create domain knowledge in a real setting (description of forged ID documents by forensic experts) with no defined vocabulary and without starting data. For both experiments, we discuss the quality of the created knowledge as well as the usability of FORMULIS for non-IT experts.

4.1 Cooking Recipes Experiment

Methodology. We selected 42 recipes taken among the top featured recipes of a well-known cooking website[5]. FORMULIS and WebProtégé were each initialized with a small base of 9 recipes extracted from the cooking website: *cookies, pizza, sandwich, bread, pizza base, salad, tartiflette, fondue, salad dressing.* Each recipe was described by its ingredients (without quantities), preparation time, cooking time, and a qualitative ranking of its difficulty and of its cost (*e.g.*, "Very cheap"). For each qualitative property, only one value was initially present. The annotators (users) were 14 volunteers among students, colleagues, IT workers, and non-IT relatives. One half used WebProtégé before FORMULIS, while the other half used FORMULIS before WebProtégé. Table 1 shows information about the annotators: the first system they used, their level of knowledge of the Semantic Web, and their previous experiences of data input interfaces. Before the experiment, only 6 annotators knew the basic concepts of the Semantic Web. In addition, 8 annotators claimed that they had already used an input data interface like spreadsheets, among them only one (User$_f$) had tested WebProtégé. No annotators had used SEWELIS or FORMULIS before.

The annotators conducted the experiment on their own, with neither supervision nor contact with other annotators. At the beginning of the experiment, each annotator received: (1) a tutorial describing how to create individuals and their properties on both systems, and (2) three recipes to enter with WebProtégé and three other recipes to enter with FORMULIS. Note that each recipe was input once in each system by two different annotators. Some recipes contained nested recipes such as sauces or bases. During the experiment, each annotator had to fill a System Usability Scale (SUS [4]) survey after using each system, rating ten assertions on a Likert scale (1: "strongly disagree" to 5: "strongly agree").

Results and interpretation: Data quality. Non-expert annotators had difficulties to distinguish the different concepts of the Semantic Web (class, individual, property) in particular with WebProtégé. Indeed, the 9 recipes present in the knowledge base before the experiment are described as instances of the `Recipe`

[5] http://www.750g.com/

Table 1. Annotators grouped by profile.

Semantic Web knowledge	**Other input interface**	**System used first**	
		FORMULIS	**WebProtégé**
None	No	User_a	User_h, User_i
None	Yes	User_b, User_c, User_d	User_j, User_k
Basic	No	User_e	User_l, User_m
Basic	Yes	User_f, User_g	User_n

class with relations to instances of the `Ingredient` class. In the final knowledge base created with WebProtégé, 31 recipes and 34 ingredients do follow the structure of the original data but 14 recipes were created as classes or as instances of `owl:Thing`, 57 ingredients were created as instances of `owl:Thing`, and 62 ingredients were created as classes or properties. The majority of invalid creations in WebProtégé were made by users with no knowledge of the Semantic Web. On the contrary, in the final knowledge base created with FORMULIS all recipes and ingredients but one follow the structure of the original data. Only one recipe and one property were created as classes. The better quality in FORMULIS can be attributed to the fact that its user interface does not easily let users depart from the original structure or do unexpected operations. Note that FORMULIS does not forbid users to do so, as the recipe created as a class shows, but only makes it more difficult, and hence makes errors less likely. It probably confused annotators that, during the creation of the value of a property, the interface of WebProtégé gives two equal choices to create either a class or a named individual, whereas named individuals are expected in most cases.

In order to evaluate, for each system, the difficulty for users to extend the model on their own, the quantity of each ingredient was given in recipe descriptions but how to describe them was not explained in the tutorial. In WebProtégé, some annotators either used ingredients as properties or OWL cardinality restrictions on recipe classes to specify quantities. In FORMULIS, User_c created a `quantity` property attached to ingredients. A positive result is that this new property was then suggested by FORMULIS, and reused by other annotators. However, the modelling is inadequate because quantities were attached to ingredients instead of to relationships between recipes and ingredients.

Globally, better quality data were created with FORMULIS than with WebProtégé. Our interpretation is that the data-driven guidance of FORMULIS helps to maintain homogeneity between new data and old data, while the lack of guidance and feedback in WebProtégé favors a wide range of mistakes. However, when a modelling error is introduced in FORMULIS, it tends to propagate from one annotator to the next. It therefore shows the necessity of a proper initialization of FORMULIS with enough examples, and expert users may be required for extensions of the models (e.g., for the quantities).

Results and interpretation: System usability. We compare the usability of the two systems as evaluated by annotators through a SUS survey for each sys-

Table 2. SUS scores relative to each experiment parameter.

		Nb. of users	**Average SUS score**	
			WebProtégé	**FORMULIS**
(i)	*Global*	14	30,7	64,1
(ii)	*WebProtégé → FORMULIS*	7	34,6	56,8
	FORMULIS → WebProtégé	7	26,8	72,1
(iii)	*SW knowledge: None*	8	25,9	61,9
	SW knowledge: Basic	6	37,1	67,1
(iv)	*Other systems: Yes*	8	32,2	58,8
	Other systems: No	6	28,8	71,3

tem. Table 2 gives the average SUS scores obtained by each system (i) globally, and according to (ii) the order in which the systems were used, (iii) Semantic Web knowledge, and (iv) experience with other systems. (i) FORMULIS was globally evaluated as significantly easier to use than WebProtégé. (ii) Annotators evaluated each system higher if they started with it, FORMULIS is still rated more than 20 points higher whatever the order of usage. (iii) Annotators with basic Semantic Web knowledge evaluated both WebProtégé and FORMULIS higher than annotators without that knowledge. Knowledge helps to reduce technical difficulties of the systems. We suppose that it is more visible on WebProtégé because technical difficulties are more vivid for beginners. FORMULIS is still rated more than 30 points higher in both cases. (iv) The gap between the perception of usability is the highest when users have not previously used another input system (more than 40 points). Even for users who had previous experience, FORMULIS is rated more than 25 points higher. In conclusion, FORMULIS is perceived as significantly easier to use by all categories of users.

4.2 Forged ID experiment

Methodology. In this experiment, the annotators are six forensic experts from the forged ID unit of the document department of IRCGN. They are experts at a national level in the forensic domain but have no prior knowledge in Semantic Web technologies. All annotators had experience with web forms or spreadsheets, and User$_1$ and User$_4$ are familiar with relational databases. User$_1$ is an IT specialist, User$_4$ is a chemist, User$_2$ and User$_5$ are handwriting experts and User$_3$ and User$_6$ are ID document experts.

Each annotator had to put into a knowledge base the description of a dozen forged Portuguese ID cards seized during a police operation. In order to provide those descriptions each annotator had to carefully examine the documents in order to assess at least eighteen description attributes, such as paper imperfections or ultraviolet reaction. The difficulty of this exercise was that the description vocabulary was not totally defined prior to the experiment and that the annotators had different knowledge background. Hence the vocabulary used to describe irregular elements might differ between annotators if it is not constrained. It is in general difficult to define *a priori* an exhaustive list of irregular elements because experts discover new elements when examining new documents.

For the evaluation, the set of annotators was split into two groups of three annotators, as shown in Table 3. Each group shared its own knowledge base. Note that in the sequel, the knowledge base of group 1 is denoted by base_1 and the knowledge base of group 2 is denoted by base_2. There were three sessions. During each session two annotators, one of each group, were providing descriptions through FORMULIS for the same documents. A session was only limited by the time that the annotators could spend on the experiment. During sessions, the two annotators could not exchange about the documents they were examining. At the first session, User_1 and User_4 were chosen as first annotators as they were the most knowledgeable in forged document detection available at that moment. Choosing experts as first annotators was deemed necessary so that the vocabulary used in the database was more likely to be consistent with domain knowledge. At the end of each session, annotators were asked to fill a SUS survey in order to provide feedback on the experiment. We also measured the time spent on each document, however there was no noticeable differences between each annotator as most of their time was spent on the careful examination of each ID document.

Results and interpretation: Data quality. The first assessed aspect of FORMULIS is how helpful it is to create quality data in a knowledge base when there are several users. Precisely, we focus on how well the use of FORMULIS is limiting the input of non-standard values inside a base such as spelling variations, synonyms or typos, *i.e.* whether it maintains the homogeneity of the base values. Looking at the data into the two knowledge bases, we note that almost no non-standard values are found in the descriptions made by the annotators. Only the first two documents created by User_1 in base_1 have non-standard values for three fields, with respect to other documents. Among those non-standard values, two are capitalization discrepancies. The other one is a wording difference, *i.e.* a signature denoted by "written with a pen" instead of "handwritten". We assume that, at the beginning of the experiment, the system was unable to make suggestions, and User_1 himself, despite being an expert, was unsure of the appropriate vocabulary. The two knowledge bases use different vocabularies for some attributes, such as "Xerography" in base_1 and "Electrophotography" in base_2 to describe the same printing technique. Those differences are explained by the different professional specialties of User_1 and User_4, respectively IT and chemistry. Despite those differences, each base is homogeneous, which is what matters for further forensic analysis. Data quality is thus maintained thanks to the guidance of suggestions. However, the first stumbles of User_1 in the first two documents and the vocabulary differences between the two knowledge bases point to the importance of the base initialization in our system.

Results and interpretation: System usability. The second assessed aspect of FORMULIS is its usability as evaluated by the annotators with a SUS survey.

Table 3. SUS scores of each annotator for FORMULIS.

Gr.	Sess.	User	Nb. of treated ID doc.	Nb. of guidance ID doc.	SUS score
1	1	User_1	13	0	42.5
	2	User_2	10	13	70,0
	3	User_3	12	23	67.5
2	1	User_4	13	0	17.5
	2	User_5	10	13	47.5
	3	User_6	12	23	87.5

Table 4. Number of documents at the beginning of a session and average SUS score per session.

Sess.	Nb. of guid. doc.	Avg. SUS score
1	0	30
2	13	59
3	23	77.5

Table 3 shows the results of the SUS evaluation, and recalls the information about the annotators and their session whereas Table 4 presents the average SUS score by session. We note that the usability rated by annotators tends to rise with the number of guidance documents in the base. As the number of documents rises, the system makes more accurate suggestions to annotators. Both User_3 and User_6 (last session users) said several times during their session that the system suggested them the values they were looking for.

4.3 Discussion

In both experiments, FORMULIS helped annotators to maintain homogeneity in the knowledge base. While the Semantic Web relies on the high flexibility of RDF, it needs ways to maintain homogeneity so as to ensure that the produced data can be easily used and shared. The first experiment showed that with the same set-up, FORMULIS could be used more easily and with better results than WebProtégé, by users with little to no training. While WebProtégé is not intended for layman annotators, the comparison with FORMULIS shows that our approach of a generic and intelligent data-driven interface is valid.

As illustrated by the errors of the first user in the ID document experiment and the problems with the quantity of ingredients in the recipe experiment, the performance of FORMULIS depends on the initialization of the knowledge base. That is known as the “cold start problem” in recommendation systems [16]. The initialization step benefits from the involvement of a Semantic Web expert in order to leave as few as possible modeling decisions to the annotators.

5 Related Work

In the Semantic Web community, a great deal of effort has been made to produce knowledge. Two different approaches for RDF authoring have been developed to sustain the Semantic Web growth: (i) approaches based on conversion from an existing source of data, and (ii) approaches allowing direct creation of a base content. On the one hand, relying on existing sources allows to obtain large amounts of data from sources such as relational databases, CSV files or formatted web pages, which do not have native graph-based structures. On the other hand, manual acquisition is necessary when there are no digital sources but requires from the contributors some knowledge of the Semantic Web technologies. Mechanisms to facilitate and guide their editions are therefore needed. In the following we focus on manual acquisition, and we discuss the pros and cons of the two categories of approaches with respect to data quality and usability.

Manual acquisition through forms and wikis. RDF editors, such as those listed in [1], aim at making it easier to create RDF data. To that end, they are often coupled with a data browser presenting the current content of the base. Users have to understand the general structure of the base and its vocabulary before editing its content. Wikidata [7] uses a wiki-like interface for crowd-sourced data edition. Users can browse content as they would in a classical wiki, but contrary to DBpedia they are directly editing RDF by defining properties around a concept. Edition is done through forms allowing the creation of new values and new properties. However, forms cannot be nested, and there are no suggestions for values. Auto-completion helps to select an existing resource but it does not even take into account the field being filled. Similarly to DBpedia, the quality of Wikidata is maintained by the constant corrections of contributors and by regular inspections from administrators. Semantic Wikis [13, 5], another kind of wiki-based editing systems, enable to enhance fulltext wiki pages with semantic annotations, and therefore enable the generation of RDF documents from those pages. The main difficulty is that a special syntax has to be learned for the semantic annotations. Semantic Wikis are most useful when the main data is textual, and when the RDF data is used as metadata. OntoWiki [9] takes the opposite approach to Semantic Wikis by using wiki pages to browse and create RDF instead of extract RDF data from them. OntoWiki aims at easing RDF data authoring by providing the accessibility of wiki pages. It is adapted to cases where there are numerous users for one particular task of knowledge engineering in a distributed environment. Contrary to FORMULIS, OntoWiki does not provide guidance during edition, only basic template and relies on the user-based wiki mechanisms to maintain data quality. WebProtégé [18] is a web-based ontology editor based on Protégé [14]. As seen in the cooking recipe experiment, WebProtégé is more adapted to users with Semantic Web training. ActiveRaUL [6] is a web form generator for RDF authoring at instance level. This system generates edition forms from a description made by an expert with the RaUL ontology, describing the various form controls (textboxes, radio buttons, *etc.*) associated with the edited ontology. ActiveRaUL has been compared to WebProtégé in an experiment with twelve users with various background regarding the Semantic Web. The experiment showed that a web form-based interface was familiar enough to users to create RDF data correctly, faster and more easily than with WebProtégé. The quality of the edited data is maintained by the constraints defined by experts.

Manual acquisition using languages. RDF data can also be directly created using languages, ranging from formal languages to natural languages. Users write down sentences that can be translated more or less immediately to RDF data. Formal languages of the Semantic Web such as the Turtle notation of RDF or SPARQL updates require no translation but demand that users master their syntax and semantics. Controlled Natural Languages [3] have been proposed as a compromise between a natural syntax and a formal semantics. They have also been proposed in combination with semantic wikis [12]. Existing systems often offer an auto-completion mechanism to avoid syntax errors. SEWELIS [10] guides users in the interactive construction of RDF descriptions, expressed in a language close to Turtle. Guidance works by suggesting classes, properties, resources, and literals to insert into the description under construction. A strength of that approach is that the suggestions are based not only on syntax but also on the existing RDF data so as to favor homogeneity across different descriptions. The user interface of SEWELIS is however tedious to use. First, the Turtle-like notation of descriptions is less familiar than form-based interfaces. Second, compared to form fields, properties are suggested in a random ordering, and have to be inserted one at a time with the risk of forgetting important properties. Our user experiments shows that FORMULIS offers a usable user interface, even for non-IT users, while benefiting from the strengths of SEWELIS-based suggestions.

6 Conclusion

In this paper, we have proposed an approach for guided RDF authoring, FORMULIS. This system allows to deal with the flexibility of the RDF data model without requiring supervision by Semantic Web experts. FORMULIS manages at the same time: (i) a user-friendly way to create RDF graphs thanks to forms, (ii) a powerful expression tool thanks to the nested forms and the possibility to add new fields at any time, as well as (iii) a way to maintain the homogeneity of values when several users fill the same knowledge base thanks to refined and dynamic suggestions.

We have conducted two user experiments. One compared FORMULIS to WebProtégé in a controlled experiment with layman users describing cooking recipes on both systems. The other experiment evaluated in a real setting the use of FORMULIS for domain expert users, forged ID experts from a forensic institution, to create domain specific knowledge with partially fixed vocabulary. Both experiments showed that even without the supervision of a Semantic Web expert, the system guided users to create good quality data, both well-structured and homogeneous. Moreover, the first experiment showed that FORMULIS was considered easier to use and less error-prone than WebProtégé. The second experiment showed that FORMULIS needs a small nucleus of data to start giving guidance and that it becomes more useful as the quantity of existing data rises.

Acknowledgement. We thank our partners of project IDFRAud, as well as the experts from IRCGN and the fourteen volunteers who participated in our experiments.

References

[1] Fakhre Alam, Shaukat Ali, Muhammad Abid Khan, Shah Khusro, and Azhar Rauf. A comparative study of RDF and topic maps development tools and APIs. *Bahria University Journal of Information & Communication Technology*, 7(1):1, 2014.

[2] Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. *Scientific American*, 284(5):28–37, 2001.

[3] A. Bernstein and E. Kaufmann. GINO - a guided input natural language ontology editor. In I. F. Cruz et al., editor, *Int. Semantic Web Conf.*, LNCS 4273, pages 144–157. Springer, 2006.

[4] John Brooke et al. SUS-A quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[5] M. Buffa, F. Gandon, G. Ereteo, P. Sander, and C. Faron. Sweetwiki: A semantic wiki. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(1):84–97, 2008.

[6] Anila Sahar Butt, Armin Haller, Shepherd Liu, and Lexing Xie. ActiveRaUL: A Web form-based User Interface to create and maintain RDF data. In *Int. Semantic Web Conf., Posters & Demonstrations Track-Volume 1035*, pages 117–120, 2013.

[7] Fredo Erxleben, Michael Günther, Markus Krötzsch, Julian Mendez, and Denny Vrandečić. Introducing Wikidata to the linked data web. In *International Semantic Web Conference*, pages 50–65. Springer, 2014.

[8] S. Ferré and A. Hermann. Reconciling faceted search and query languages for the Semantic Web. *Int. J. Metadata, Semantics and Ontologies*, 7(1):37–54, 2012.

[9] Philipp Frischmuth, Michael Martin, Sebastian Tramp, Thomas Riechert, and Sören Auer. OntoWiki – an authoring, publication and visualization interface for the data web. *Semantic Web*, 6(3):215–240, 2015.

[10] Alice Hermann, Sébastien Ferré, and Mireille Ducassé. An interactive guidance process supporting consistent updates of RDFS graphs. In *Knowledge Engineering and Knowledge Management*, pages 185–199. Springer, 2012.

[11] Carlos A. Hurtado, Alexandra Poulovassilis, and Peter T. Wood. Query relaxation in RDF. *J. Data Semantics*, 10:31–61, 2008.

[12] K. Kaljurand and T. Kuhn. A multilingual semantic wiki based on attempto controlled english and grammatical framework. In *The Semantic Web: Semantics and Big Data*, pages 427–441. Springer, 2013.

[13] Markus Krötzsch, Denny Vrandečić, Max Völkel, Heiko Haller, and Rudi Studer. Semantic wikipedia. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(4):251–261, 2007.

[14] N.F. Noy, M. Sintek, S. Decker, M. Crubezy, R.W. Fergerson, and M.A. Musen. Creating semantic web contents with Protege-2000. *Intelligent Systems, IEEE*, 16(2):60–71, 2001.

[15] G. M. Sacco and Y. Tzitzikas, editors. *Dynamic taxonomies and faceted search.* The information retrieval series. Springer, 2009.

[16] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In *Annual Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, SIGIR '02, pages 253–260, New York, NY, USA, 2002. ACM.

[17] SPARQL 1.1 query language, 2012. W3C Recommendation.

[18] Tania Tudorache, Csongor Nyulas, Natalya F Noy, and Mark A Musen. WebProtégé: A collaborative ontology editor and knowledge acquisition tool for the web. *Semantic web*, 4(1):89–99, 2013.