Web of Data Evolution by Exploiting Agent Based-Argumentation

Fatma Chamekh, Danielle Boulanger, Guilaine Talens

Web of data evolution by exploiting agent based-argumentation

Fatma Chamekh[1], Danielle Boulanger[1], Guilaine Talens[1]

University of Lyon-Jean Moulin University-Magellan Research center,Lyon,France
fatma.chamekh@univ-lyon3.fr, danielle.boulanger@univ-lyon3.fr,
guilaine.talens@univ-lyon3.fr

Abstract. Sharing knowledge and data coming from different sources is one of the biggest advantage of linked data. Keeping this knowledge graph up to date may take in account both ontology vocabularies and data since they should be consistent. Our general problem is to deal with web of data evolution in particular: We aim at assisting user in a such complex process. In this research work, we propose an agent based-argumentation framework to help user linked data changes. We assign for each agent a goal to acheive as process step. We rely upon communication-argumentation potential of the agent to harmonise changes with consistency

Keywords: Linked Open Data; Web of Data evolution; multi-agent system; consistency; ontology.

1 Introduction

Since Tim Berners-Lee presented the new issue of semantic web, linked data (web of data) becomes a promising research field. The Term "*linked − data*" refers to a set of best practices for publishing and connecting structured data on the web [1] :

- URIs to denote things.
- Use HTTP URI so that these things can be referred to and look up (dereferenced) by people and user agent.
- Provide useful information about the thing when its URI is dereferenced, leveraging standards such RDF and SPARQL.
- Include links to other related things (using their URIs) when publishing data on the web.

Even sence, the meaning of data can be encoded using ontologies represented in RDF[1] , whose semantics may be defined within the RDF Schema (RDFs[2]) and Web Ontology Language (OWL[3]) standards. Following those principals,

[1] http://www.w3.org/TR/2014/REC-rdf-schema-20140225/
[2] http://www.w3.org/TR/rdf-schema/
[3] http://www.w3.org/TR/owl2-primer/

there has been much effort from both academia and industry to build a multi-domains and multi-sources connected datasets. Through these knowledge graphs, fragmented information has been aggregated to facilitate automatic reuse. The amount of data interlinked had been increased to offer a great potential to build strong knowledge base, smart products and services that create new value.
Facts are recorded in the web of data which is a noteworthy advantage rather then web document [2].This becomes the bases of publishing and discovering knowledge. However, keeping knowledge up to date is fundamental to guarantee a long-term usability. To do this, we need to feed continuously the dataset.The dataset evolution is still a tricky and a time consuming task, since it requires the intervention of human experts, whose role is to analyse the changes, to connect the new triples in the right positions and give them a structural and semantic consistency before they can be exploited. This makes changing a crucial step, where some knowledge may still remain noisy or missing.
The agent specifications can facilitate the process of dataset evolution : the idea we propose here is not only that agent technology is one of the most useful solution to cope with a complex problem such as web of data's evolution, but also the agents are autonomous in the sense that they have the ability to decide themselves which goals they should adopt and how this goals should be acheived using their knowledge base. Beside that, they need to communicate to accomplish their tasks and resolve differences of opinion and conflicts of interest. We highlight the agents negotiation capability by argumentation for working together to find solution and construct proof. With such agent potential, it could be possible to automatise the process of web of data evolution, or at least assist the user in such approach.
In this paper, we aim to handling with the complexity of linked data changes by according to each agent a role (a process step) and exploiting agent argumentation to drive the consistency problem. We assume that we use OWL/RDF/RDFs constraints various rules to detect inconsistencies and we work in linked data closed domain. We assume that these rules are encapsulated in agent for assisting it to decide about the concerned change. Agents use argumentation to reach consensus about the best change alternative without inconsistency in an a priori way. Relatively to this goal, how the agent-based argumentation gives help to user to drive the web of data evolution?
The remainder of this paper is as follows. We first present the difference between the ontology evolution and web of data evolution. Then, we compare the proposed approach with other approaches adopting the argumentation for inconsistency in Dbpedia and ontology alignement, and others proposing a framework for web of data evolution (section 2). Secondly, we present our approach: The system architecture, the structure of each agent and argumentation. Finally, we conclude by sum up the ideas and propose some future works.

2 Problem definition

The scenario we use to illustrate our problem involves the general medicine domain.
The text below is an extract from a medical report. It describes a clinical case. It deals with specific subjects (asthma and obesity) by presenting the experiments, results and publications.
peak expiratory flow (PEF) variability, asthma and forced expiratory volume in 1 second (FEV1) or PEF of expected values ;treatment with beta2-agonists combined with high doses of inhaled corticosteroids (ICS) with additional therapy and oral corticosteroids.
Based on those reports, we have been created Ontologos RDF dataset. The figure 1 shows an extract of Ontologos dataset. Data are grouped according to four main concepts : OL:diseases, OL:symptom, OL:scientificpaper, OL:treatment.

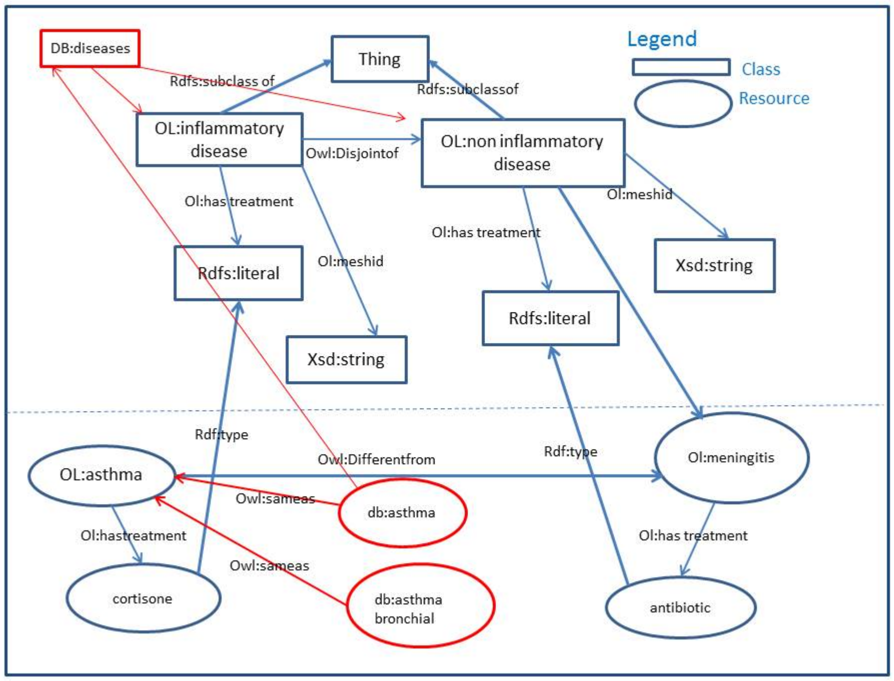

Fig. 1. The extract of Ontologos dataset.

Evolving Ontologos dataset : The number of clinical reports increases. To evolve the dataset, we beleive that we could extract knowledge from those reports and the LOD. For this end, we use DBpediaSpothlight service. The result

is a triple $< C, rdf : type, R >$.
Our assumption is that changes do not happen by pure luck, but un underlying reason make them related to other existing triples. Finding underlying reason is defined by measuring the similarity between two triples. Althougth, it is not enough to decide about the acceptability of changes since we expect a consistent result.
Here, the challenge is, how can we evolve and keep consistency of Ontologos dataset?

3 Proposed approach

To deal with the complexity of the dataset evolution issue, we use a multi-agent system. Our motivation is that agents work as team [3]. Each one have a specific task. Agents could negotiate to make a decision about a proposition of change.

3.1 Framework description

In our system, we make use of three different types of agent as shown in Figure 2.

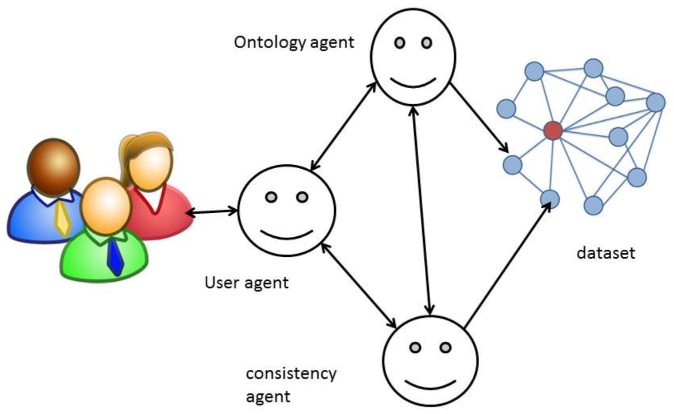

Fig. 2. The multi-agent system

1. User agent is the agent interface that provides interactions among user and system and proposes to the ontology agent the new item annotated from documents with its different URIs.
2. Ontology agent analyses the possible position of the new statement and its URI. To do this the agent has an available knowledge base. This one is a set of rules based on similarities measures (syntactic and semantic).
3. Consistency agent studies the effect of this change upon the ontology and data consistency. Checking consistency is made by logic description rules encapsulated in the agent.

The effective collaboration between the different agent types contributes greatly to the efficiency of the system performance. To draw the different interactions between agents we use the UML sequence diagram shown in Figure 3. From this

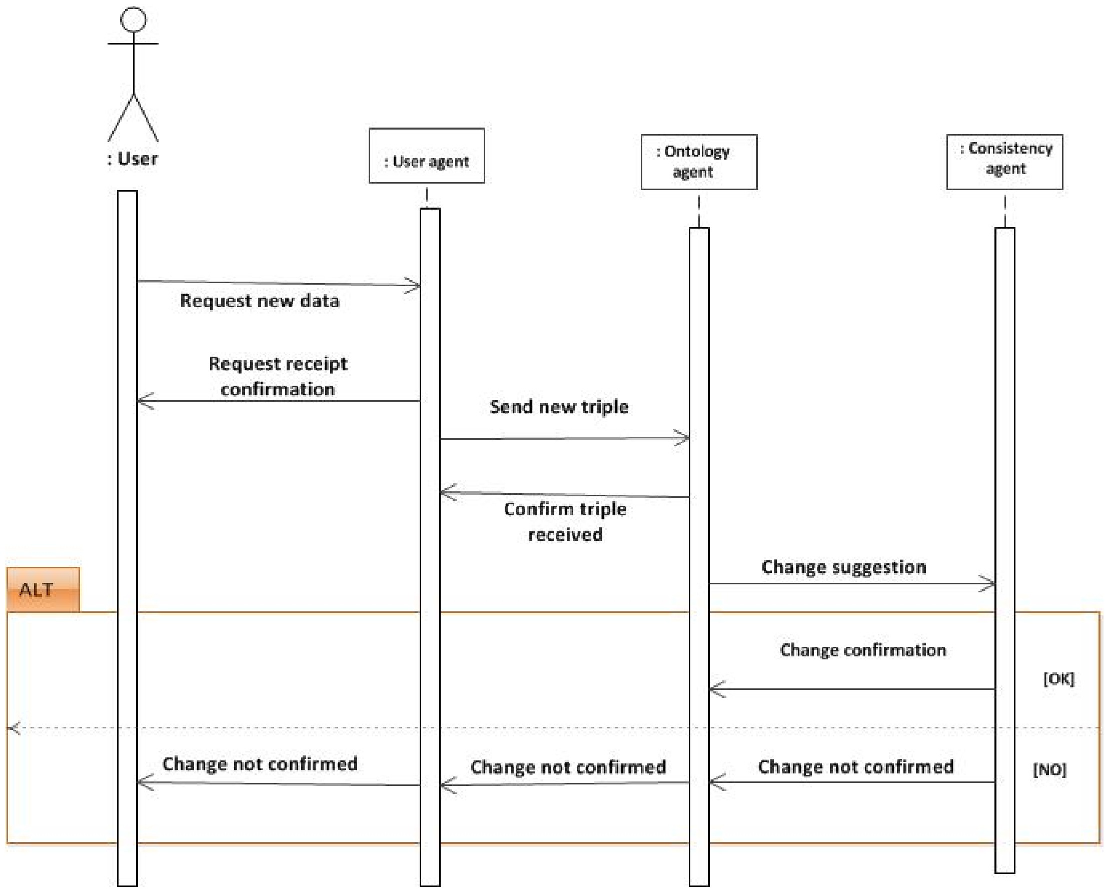

Fig. 3. Sequence diagram that models inter-agent communication.

diagram, we can see that the user makes an initial request to the user agent to add data as result of annotation process. The user agent responds to the user (request receipt confirmation) indicating that it has received the request. Once the user agent has received this data, it sends the data to the ontology agent. As soon as the ontology agent gets the data from user agent, it sends a message back to the user agent confirming that it has received the data.The ontology agent computes syntactic and semantic similarities between base ontology and the new data. To do this, the latter uses its internal knowledge base. Once ontology agent gets the similarities results, it is then able to send a suggestion of changes that could be made. The consistency agent, uses that knowledge to detect inconsistency regarding this change. If there is no inconsistency, the consistency agent sends its confirmation to the ontology agent. Then it creates the new change. If there is an inconsistency, the ontology agent informs the user that changes could not be made.
In order to do their job properly, each agent should be organized as a set of behaviors, knowledge base. We present the structure of each agent in the following sections.

User Agent The user agent interacts with the user. It gets the changes requested by the user, analyses it by specifying the concerned data and sends them to ontology agent. Each user agent manages one change related to one user. To coordinate those tasks, we present the general behavior by algorithm 1.

Algorithm 1: user agent general behavior

```
Require: changes
Ensure: T= $< Subject, Predicate, Object >$
  perceive()
  Analyses changes ()
  Sends changes to ontology agent
  if T exists then
    informs user that the operation will be dropped
  else
    informs user that the operation has been accepted successfully
  end if
  stop agent()
```

Ontology Agent The ontology agent is dedicated to put the new triples (new data) in the most suitable position in the initial graph. There is one agent ontology for each new change triple.To depict all tasks, the general behavior algorithm is presented in algorithm 2.

Algorithm 2: ontology agent general behavior

```
Require: T= $< Subject, Predicate, Object >$
  perceive()
  Similarity index = compute similarities ()
  if $Similarityindex \geq 0.5$ then
    propose position()
    request Suggestions S=$s1, s2, ..., sn$ to consistency agents
  else
    no possible changes
    informs user agent
  end if
  stop agent()
```

Positioning the new triple is based on computing syntactic and semantic similarities between the new entities and the existing ones. We use the Levenshtein distance [4] to compare label of each entities and Rada distance [5] to calculate the number of arcs on the shorter path between two terms :
dist(t1,t2)=length(t1,lest(t1,t2))+length(t2, lest(t1,t2)). We can define the distance between two RDF triples using this distance as a sum of relations between two predicates, two subjects, and two objects:
dist(triple1, triple2) = dist(subject triple 1,subject triple 2) + dist(predicate triple1,predicate triple2) + dist(object triple1, object triple2).
According to the similarity index, the ontology agent suggests a suitable position using rules encapsulated on its knowledge base.

Consistency agent Keeping consistency of the graph knowledge is the consistency agent's role. To acheive its goal, it uses the rules saved in its knowledge base. Those rules are has been identifying under RDF/RDFs/OWL constraints. Each consistency agent handles changes suggestion. We present the general behavior by algorithm 3.

The agents presented above have to reach agreement concerning a proposition of change. Since their knowledge base are different, their perceptions of the environment are also different. Hence, an agent could decide depending on its preferences whether to accept or refuse a proposition of change. To reconile those points of view, agents should negotiate through exchanged arguments. In the following section, we describe the argumentation approach.

Algorithm 3: consistency agent general behavior

```
Require: T= $< Subject, predicate, object >$
  perceive()
  verify inconsistency ()
  if no inconsistency then
    process change()
  else
    proposes additional changes ()
    process to the user
  end if
  Stop agent()
```

4 The argumentation approach

Argumentation represents the study of views and opinions expressed by humans with the goal of reaching a conclusion through logical reasoning [6]. Several models were proposed to capture the essence of informal argumentation in different settings. We start with the presentation of Dungs work [7].

Definition 1 : An Argumentation Framework (AF) is a pair AF = $< AR, A >$, where AR is a set of arguments and $A \subset AR \chi AR$ is the attack relationship for AF. A comprises a set of ordered pairs of distinct arguments in AR. A pair $< z, y >$ is referred to as z attacks y. We also say that a set of arguments S attacks an argument y if y is attacked by an argument in S.

Dung framework can be simply represented as a directed graph whose vertices are the arguments and whose edges represent conflicts among elements as a binary relation called attack. In Dung's framework, all arguments have an equal strength and attacks always succeed [7]. This could be possible when dealing with deductive arguments. However, in a specific context knowledge consistency, an objection can still be raised about the lack defined consensus for dimensions and metrics. The need to introduce persuasiveness of each arguments leads to the proposal called A Value-Based Argumentation Framework (VAF) [8]. This one considers the preferences between arguments where an argument can provide reasons.

Definition 2 : A Value-Based Argumentation Framework (VAF) is defined as $< AR, A, V, n >$, where (AR,A) is an argumentation framework, V is a set of k values which represent the types of arguments and n:AR→ V is a mapping that associates a value (x)∈ with each argument x∈ AR.
The strengths of arguments is related to their different audiances with different interestes and preferences.

Definition 3 : An audience for for VAF is a binary relation R $\subset$ VxV whose (irreflexive) transitive closure. R^* is asymmetric, at most one of $(v, v'), (v', v)$ are members of R^*. We say v_i is preferred to v_j, denoted $v_i \succ v_j$, if $v_i, v_j \in R^*$. So, we take into account that different quality asessment frameworks or modules can have different perspectives to dataset according to the dimensions and metrics. Since the set of arguments and against arguments have been produced, it is necessary to consider which of them should be accepted. Acceptability of argument is defined as following :

Definition 4 : Let $< AR, A, V, n >$ be an argumentation framework. For R and S, subsets of AR, we say that:

- An argument s $\in$ S is attacked by R if there is some r $\in$ R such that $< r, s >$ $\in$ A.
- An argument x $\in$ AR is acceptable with the respect to S, if for every y $\in$ AR that attacks x, there is some z $\in$ S that attacks y.
- S is conflict free if no argument in S is attacked by any other argument in S.
- A conflict free set S is admissible if every argument in S is acceptable with the respect to S.
- S is a preferred extension if it is a maximal (with the respect to set inclusion) admissible subset of AR.

The key notion is the preferred extension which presents a consistent position in the AF. According to the preferred extension, [8] introduces the notion of subjective and objective acceptance as follows :

Definition 5 : Given VAF $< AR, A, V, n >$, an argument s $\in$ AR is subjectively acceptable if s appears in the preferred extension for some specific audiences but not all. An argument s $\in$ AR is objectively acceptable if s appears in the preferred extension for all specific audiences.

4.1 The argumentation for consistency

In this paper, we focus on arguments about data consistency. We define the argument's concept as follows :

Definition 6 : Let an argument $x \in AR$ is a triple $x =< G, c, \sigma >$ where :

- G is a correspondence $< changes, consistency >$
- c is the "grounds" justifing a "primafacie" belief that the correspondence exists, or not.
- σ is one of (+,-) depending on whether the argument is that G exists or not.

The interactions between arguments is based on the notion of attack. An argument x is attacked by the assertion of its negation$\neg x$, namely the counter argument, defined as follows:

Definition 7 : An argument y$\in$ AF rebuts an argument x$\in$ AF, if x and y are the arguments of the same proposition of change but with different sign, e.g if x and y are in the form of $x =< G, c, + >$ and $y =< G, c, - >$, x counter-argues y and vice versa.
The arguments are identified on the underlying of RDF/OWL language. Therefore, the grounds justifying correspondances can be extracted from the knowledge in the RDF repository. Our classification of the grouds justifying change proposition is the following :

- Syntactic (SY) : the labels of entities share more or less syntactic features.
- Semantic (S) : the entities share some properties (dataproperties)
- Logic (L) : the RDF graph is conform to the logic rules.

In our framework, we use the types of arguments described above for the VAF, hence $V = \{SY, S, L\}$.

For example, the audience may specify that logic arguments are prefered to semantic arguments or vice versa. Note that this could vary according to the RDF dataset to be evolved. In case that the RDF datasets are published in the LOD cloud, the logic arguments are given more weight, compared to the private RDF dataset where there is important to have the suitable vacabularies and relationships among them. Table 1 represents a simple reason for the justifying of RDF dataset changes. The table represents a set of argumentation schema, the instantiation of which include AR. Attacks between these arguments arise when we have arguments for the same proposition of change but with conflicting values of σ, thus yielding attacks that can be considered symmetric.

For example, given a candidate change $ch =< c, c', -, =>$ between two concepts c and c'. The first one is coming from the extraction results (from text) and the second one is the existing concept in the RDF dataset. An argument for accepting the change "ch" may be the label of c and c' are synonymous. an argument against is that may be that there is no common Dataproperties between c and c'.

The proposition of change	σ	**ground**	**description**
$< r, owl : sameAs, r' >$ $< r, owl : differentFrom, r' >$	+	$T(r) = T(r') = rdfs : literal$	The both resources r and t' are literal
	+	V(R)=V(R')=$[0-9]+\|o=[a-z]+$	The both resources are either numeric or literal value
	-	URI(r)=URI(r')	The both resources have a similar URI
	+	$URI(R) \neq URI(R')$	The both resources have different URI
	-	$CT(R') = \emptyset$	The resource R' doesn't have a class type
	+	CT(R')={C,C',..}	The resource R' has more than one class type
	-	P(R')=0	The resource R' is isolated
	+	S(owl:DataProperty)={R,R'}	The resources Rand R' could be the domain of the same DataProperty
	+	O(owl:ObjectProperty)={R,R'}	The resource R and R' could be the range of the same ObjectProperty
	+	S(owl:ObjectProperty)={R,R'}	The resources R and R' could be the range of the same ObjectProperty
	+	Ctype(R)inclutCtype(R')	The resources R and R' have a common class type
	-	$< C, owl : Disjointclass, C' >$	C and C' are disjoint class
$< c, owl : equivalentClass, c' >$ $< c, rdfs : subClassof, c' >$ $< c, rdfs : subClassof, thing >$	+	$URI(C) \neq URI(C')$	The both classes have different URI
	-	URI(C)=URI(C')	The both classes have the same URI
	+	$D(R1) \in \{C, C'\}$	C and C' could be a domain of the resource R1
	+	$R(R1) \in \{C, C'\}$	C and C' could be a range of the resource R1
$< c, rdfs : subClassof, c' >$	-	$DT(C) \subset DT(C')$	C and C' have a common dataproperty
	+	$DT(C) \not\subset DT(C')$	C and C' have a non- common dataproperty
	-	$SUPC(C') = \emptyset$	There is no super-class for c'
$< c, owl : equivalentClass, c' >$	+	$DT(C) \subset DT(C')$	C and C' have a common dataproperty
	-	$DT(C) \not\subset DT(C')$	C and C' have a non- common dataproperty

Table 1. The argument schema

4.2 Arguments generation process

In this section, we descride how agents generate the arguments. Each agent A has an access to its knowledge base:

Definition 8 : An agent Ag is caracterised by a 3-tuples $< kb, VAF, pref >$, where kb is the internal knowledge base, $VAF =< AR, A, V, n >$ is the value-based argumentation framework of the agent Ag, pref is a private threshold value represents preferences over V.

A multi-agent system (MAS) is a set of agents $A = \{Ag_1, Ag_2, ..., Ag_n\}$. The agents share a set of arguments. Those arguments could have the same values. The preferences selected by an agent depend on its context and situation. Akey feature of this context is the RDF repository. This one depends on its Logic features (eg. depth of subclass), its semantic feature (eg. a common dataproperty between class), and its syntactic features (eg. some resources have the same label). The analysis of one of one of those caracteristics is made by ontology agent and consistency agent. An agent can determine its prefrences based on the caracteristics of the dataset. For exemple, selecting a preference for logic consistency if the dataset is deficient in logic constraints.

In our framework, the agents generate the arguments and counter-arguments using theses preferences. We assume a set of agents commiting to the existing RDF dataset storing the different propositions of changes. Different propositions of changes are proposed to ontology agent. This one uses its knowledge base to propose a relationship between the new term and the existing one. Following the proposition of change, the consistency agent analyses the effect of the change upon the data and ontology consistency. In order toenable the agents tocome to agreement on a suitablechange without requiring a complete modification of the dataset. The consistency agent specifies which of the elements from the RDF repository is involved. In this case the ontology agent and consistency agent should be agree concerning a proposition of change.To generate arguments, agents use the algorithm 4.

Given an agent $Ag =< Kb, VAF, pref >$ and a candidate change "ch" with a set of justifications G. First, the agent evaluates the acceptability of "ch". A change "ch" is accepted by an agent Ag, if it exists justifications G for ch that correspond to the highest performence pref. Consequently, the agent generates a set of arguments $x =< G, c, + >$. In the other case, the agent rejects the change by generating a set of arguments against $x =< G, c, - >$. The model of the communication between the ontology agent, the consistency agent, and the dataset is shown in the table 2. This model defines the way in which agents cooperate. The model presented in the table 2 describes the different steps of the argumentation process.

Algorithm 4: generation of arguments

```
Require: changes
Ensure: A set of agents $Ag = \{Ag1, Ag2, ....Agn\}$
  for $Ag =< Kb, VAF, pref >$ do
    for change ch do
      if $pref \in G$ then
        agent generates arguments $x =< G, c, + >$
      else
        agent generates arguments $x =< G, c, - >$
      end if
    end for
  end for
```

1	Ontology agent calculates similarities between new triples and old one. The agent knows which element of the dataset will be involved.
2	Ontology agent knows that consistency agent exists
3	Ontology agent sends the proposition of change to the consistency agent.
4	Both agents known the involved element in the dataset.
5	Both agents exchange their argument.
6	Both agents instantiate their argumentation framework.
7	Both agents calculate their preferred extensions.
8	Both agents determine the agrees change by exchanging the preferred extensions.
9	Both agents confirm the change.

Table 2. Agents communication

4.3 Accepted and considered arguments

In VAF, there is a unique non-empty prefered extension without the respect to specific audience. An agent may have multiple prefered extensions either because no preference between two values in the cycle has been expressed, or because a cycle in a single value exists. In our domain, where many attacks are symmetric, two cycles are frequent and an general an audience may have multiple preferred extensions. Thus, given a set of arguments justifying mappings organised into an argumentation framework, an agent is able to determine which mappings are acceptable by computing the preferred extensions with the respect to its preferences. If there are multiple preferred extensions, the agent must commit to the argument to the arguments presented in all prefered extensions, but has some freedom of choice with the respect to those in some but not all of them. Those arguments are divided on three sets : desired arguments, present in all preferred extensions, optional extensions, present in some but not all preferred extensions, and rejected arguments, present in none preferred extensions. If we have two agents belonging to differents audiences, these way differ. According to the above considerations, we thus define an acceptable change and considered change as follows. An acceptable change is the set of correspondences supported by those arguments which are in every preferred extension of each agent. A considered change extends the agreed change with those proposition of change supported by arguments wihch are in some preferred extension of every agent. To generate the both acceptable and considered change, the agents follow the algorithm 5.

Algorithm 5: Acceptable and considered change

```
Require: Argumentation framework $< AR, A, V, n >$; arguments
  $< B, P, \sigma >$;audience $R_i$; proposition of change p
Ensure: The acceptable proposition of change(PA);The considered
  proposition of change(PC)
  $PC = \emptyset$
  $PF = \emptyset$
  For all audiences $R_i$ compute preferred extension
  for $R_i$ do
    For all arguments values coming from arguments schema (AS)
    for AS do
      $P_j(< AR, A, V, n >, R_i), j >= 1$
      $P_k(R_i) = \cup P_j(< AR, A, V, n >, R_i)$,$k >= 0$
      PA gathers all the desired arguments
      $PA_{arg} = x \in P_k(R_i)$, such as $k >= 1$ and $i >= 0$
      for $x \in PA_{arg}$ do
        if $x =< B, P, + >$ then
          $PA = PA \cup p$The proposition is rejected
        end if
        PC gathers all the optional arguments
        if $\exists p \in P$ such as$p \notin PA$ and p is not rejected then
          $PC_{arg} = x \in P_k(R_i)$ , such as $k >= 1$ and$i >= 0$
        end if
        for $x =\in PC_{arg}$ do
          if $x =< B, P, + >$ then
            $PC = PC \cup p$
          end if
        end for
      end for
    end for
  end for
```

5 An illustrative example

To illustrate the approach presented in this paper, we introduce Alice : a hypothetical end-user of an application for searching and browsing the Ontologos dataset. Alice loads some interesting data about inflammatory and non inflammatory diseases. She tried to load more external data (coming from another datasets), but she didn't find them. For this goal, Alice wants to evolve the Ontologos dataset from her medical reports and Linked Open Dataset (LOD) Cloud.
For this end, Alice annotates the medical reports using DBpedia Spotlight[4]. This tool allowed Alice to automatically annotating mentions of DBpedia resources in text. DBpedia Spotlight recognises that the names of concepts or entities have been mentionned and subsequently matches those names to unique identifiers. Besides common entity classes (i.e. People, Locations, Organisations), Spotlight also detects concepts from the 320 classes in the DBpedia Ontology.
Alice gets the following triples as results of annotation task :

- Resource 1 :$< db : Asthmabronchial, rdf : type, >$
- Resource 2 : $< db : asthma, rdf : type, db : diseases >$

To add those triples to the Ontologos dataset, the user agent sends the resources to the ontology agent. This one computes similarity between the new triples and those from the existing dataset. The similarity index between them are :

- dist1(db:Asthma et ol:asthma)=1
- dist2(db:diseases et ol: inflammatory diseases)=0,55
- dist3(db:diseases et ol:non inflammatory diseases)=0,55
- dist4(db:asthmaBronchial et ol:asthma)=0,6
- dist5(db:diseases et ol: inflammatory diseases)= 0,85
- dist6(db:diseases et ol:non inflammatory disease)= 0.85

According to these similarities indexes, the ontology agent suggests the following propositions of changes and sends them to the consistency agent :

- prop1:$db : asthma, owl : sameAs, ol : asthma$
- prop2: $db : asthmaBronchial, owl : sameAs, ol : asthma$
- prop3:$ol : inflammatorydiseases, rdf : subClassof, db : diseases$
- prop4:$ol : noninflammatorydiseases, rdf : subClassof, db : diseases$

The consistency agent cheks the effects of this change. To argue about one or more changes the ontology agent and consistency agent negotiate arguments for each proposition of change. We assume that the ontology agent selects the audience R1 which prefers syntactic to logic. The consistency agent selects the audience R2 which prefers logic to syntactic. The arguments and counter-arguments generated are shown in Table 3, that shows each argument, labeled with an identifier ID, its type V, the attacks A that can be made on it by opposing arguments.

[4] https://github.com/dbpedia-spotlight/dbpedia-spotlight

ID	Arguments	A	V
A	$< URI(db : asthma) \neq URI(ol : asthma), prop1, + >$		SY
B	$< URI(db : asthmaBronchial) \neq URI(ol : asthma), prop2, + >$	E	SY
C	$< CT(db : asthma) = (Db : diseases), prop1, + >$	F	S
D	$< CT(ol : asthma) = (ol : inflammatorydiseases), prop1, + >$	G	S
E	$< CT(Db : asthmaBronchial) = \emptyset, prop2, - >$	B	L
F	$< (Db : diseases), rdfs : subclassof, , prop4/prop3, - >$	I,J	L
G	$< ol : inflammatorydiseases, rdf : subclassof, thing, prop3, - >$	F	L
H	$< ol : noninflammatorydiseases, rdf : subclassof, thing, prop4, - >$	F	L
I	$< URI(Db : diseases) \neq URI(Ol : inflammatorydiseases), prop3, + >$	F	SY
J	$< URI(Db : diseases) \neq URI(Ol : noninflammatorydiseases), prop4, + >$	F	SY

Table 3. Arguments for and against the propositions of changes

- **The argument A** supports the proposition of change prop 1 because the URI(db:asthma) is different from URI(ol:asthma).
- **The argument B** states the proposition of change prop 2 because the URI(db:asthmaBronchial) is different from the URI(db:asthma).
- **The argument C** supports the proposition of change prop 1 because (Db:diseases) is a class type of (db:asthma).
- **The argument D** supports the proposition of change prop 1 because (ol: inflammatory diseases) is a class type of the resource (ol:asthma).
- **The argument E** is against the proposition of change prop2 because the resource DB:asthmaBronchial doesn't have a class type.
- **The argument F** is against the proposition of change prop3 and prop4 because the class (Db:diseases) doesn't have an upper class.
- **The argument G** is against the proposition of change prop3 because the upper class of ol: noninflammatory diseases is Thing.
- **The argument H** is against the proposition of change prop4 because the upper class of ol: noninflammatory diseases is Thing.
- **The argument I** supports the proposition of change prop3 because the URI(Db:diseases) is different from the URI(Ol: inflammatory diseases).
- **The argument J** supports the proposition of change prop4 because the URI(Db:diseases) is different from the URI(Ol: noninflammatory diseases).

Based upon these arguments and the attacks, each agent can build the argumentation framework shown in Figure 4. The nodes present arguments (labelled with their ID) with respective type value V. The edges represent the attacks A, and the direction of the edges represents the direction of the attack.

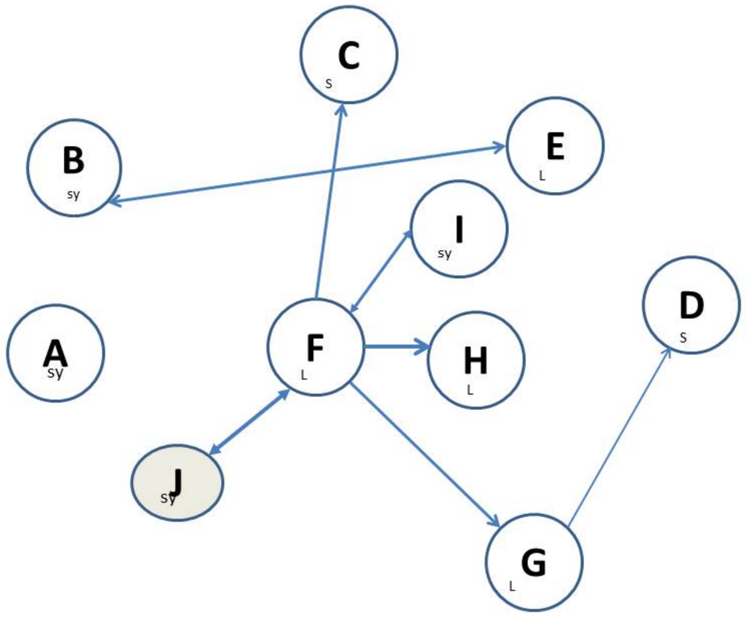

Fig. 4. Argumentation framework

The preferred extension of R1 are {A,D,G,F,H} and {A,G,F,C,H}. The preferred extensions of R2 are {A,E,B,F,G,D} and {A,C,F,G,D,E,B}. The arguments that are accepted by both audiences are {A,D,F,H,C,G,B,E}. Hence, the accepted propositions of changes are prop1 and prop2.
The system requests Alice to validate the proposition of changes. To finally conclude, Alice brows again the Ontologos dataset, and she finds the new resources.

6 Related works

Dynamic knowledge, in particulary ontology evolution has attracted the interests of several researches. Different approaches are designed as a multi-step process to adapt the changes through the time [9]. Those works can be grouped into user-driven and external knowledge approaches.
User-driven approaches are focused on dealing with user assistant to acheive the ontology evolution process. Generally, those approaches allow user or ontology engeneering to modify directly the ontology. Each of them proposes a different solution to keep the ontology consistency and artefacts propagation.

In [9], authors use a precondition and postcondition for the control and resolution of consistency. For [10] the consistency pattern is the most fitting solution to express all the features. Furthermore, conserving the homogeneity between artefacts as mentionned in, is depicted by following the dependencies between annotation and ontology, and the dependencies between documents and ontology in [11].
The second group involves external background knowledge to feed the ontology by a new statement with the respect to the initial ontology [12]. Some works such as [13] are focused on building and evolution ontology from texts. In [13] the agents give help by using their capacity of adaptation to build an ontology as agent network composed by agent term and agent concept.
To summarise, all the frameworks presented above are based on a language of changes, which describes the semantics of change operations. This latter consists of three low-level operation : Add(x), Delete (x), Modify(x). A-Box and T-box entities are concerned singly.
In the Linked data context, the data have gained a lot of attentions from developpers, publishers and users. According to the linked data paradigms, ontology /schema is usually an association of existing vocabularies (i.e DBpedia, FreeBase, Bio2RDF..etc). As RDF database the main structure is a triple (Subject, predicate, object), both of subject and object should be a resource which might be a multiple type. Even sense, [14] have presented an algorithm to improve the quality of linked data using statistical distributions. The first algorithm adds missing type statement and the second one identifies faulty statement. The LOD (Linked Open Data) Stack 2 [15] is an integrated distribution of aligned tools which support the whole life cycle of linked data [16] . To deal with modifications in both data and ontologies, a set of strategies is proposed. These approaches are expected to tackle issues such as: synchronisation problem, citation problem, quality of the data set and preserving long-term access. The Diachron platform [2] manages the preservation of evolving linked data ecosystems. In particular, the evolution module is responsible for detecting, recording and managing changes of LOD by comparing different versions of LOD dataset. The framework we propose has common points with this line of works, i.e the changes and preserving linked data consistency, but the previous approaches are tackled this problem in an a posteriori way, that might be difficult to verify all inconsistency features at least schema/ ontology. So, verifying consistency in an a priori way could be more efficient, in the sense that we detect at least logical and structural inconsistency, and we avoid its propagation.
Following this research line, [17] classify inconsistencies in DBpedia Language specific chapters based in the linguistic phenomena (i.e identity, acronym, identity, hyponymy..). To detect and reconcile inconsistencies coming from different languages specific DBpedia chapter, the authors used an argumentation theory. This approach adopts a bottom-up technic to detect inconsistencies. Finally, the idea of using agent-based argumentation has been exploited in ontology alignment [18] to facilitate the communication among agents which have a different vocabulary or ontology. In this paper, the argumentation is the communication

way between agents to reach decision about changes. The arguments are the types of changes and changes consequences, both of them should be reconciled to avoid inconsistency.

7 Conclusion

In this paper, we have outlined a framework to evolve the web of data. This framework is based on the agent technology. The agents handle the complexity of the linked data changes. Each agent has a role. For this aim, we distinguish three agent types : user agent, ontology agent and consistency agent. To agree upon a proposition of change, the ontology agent and consistency agent use the argumentation process. Based on the agent's knowledge base and the agent's preferences, the proposition of change could be accepted or rejected. This will give the agents the ability to carry out the modification of knowledge graph in a smoothly way. We believe that this approach makes easier the communication between agents. Currently, the preferences are limited to a few consistency criterea.

In the future works, we will extend these preferences to the data quality dimensions. We aim to argue about the data quality of the dataset. Another perspective that we will consider in this research work is the dataset versionning. Each change leads to create a new version of dataset. To keep track of each change, we propose to annotate each modified resources. To handle those modifications on a multi-user context, we propose a set of rules for conflict resolution.

References

1. Heath, T., Bizier, C.: Linked Data: Evolving the Web into a Global Data Space. Morgan & Claypool (2011)
2. Papastefanatos, G., Stavrakas, Y.: Diachronic linked data: Capturing the evolution of structured interrelated information on the web. Ercim news **52** (january 2014) 35–37
3. Ferber, J., Gutknecht, O.: A meta-model for the analysis and design of organizations in multi-agent systems. In: Proceedings of the Third International Conference on Multiagent Systems, ICMAS. (1998) 128–135
4. Levenshtein, V.I.: Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Soviet Physics Doklady **10** (1966) 707
5. Rada, R., Mili, H., Bicknell, E., Blettner, M.: Development and application of a metric on semantic nets. IEEE Trans. Systems, Man, and Cybernetics **19**(1) (1989) 17–30
6. Schneider, S., Groza, T., Passant, A.: A review of argumentation for the social semantic web. Semantic Web **4**(2) (2013) 159–218
7. Dung, M.D.: On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming and n-person games. Artificial intelligence **77**(2) (1995) 321–358

8. Bench-Capon, T.: Persuasion in practical argument using value-based argumentation frameworks. Journal of Logic and computation **13** (2003) 429–448
9. Stojanovic, L.: Methods and tools for ontology evolution. PhD thesis, Karlsruhe Institute of Technology (2004)
10. Djedidi, R., Aufaure, M.: *ONTO-EVO*^{a}*L* an ontology evolution approach guided by pattern modeling and quality evaluation. In: Foundations of Information and Knowledge Systems, 6th International Symposium, FoIKS 2010, Sofia, Bulgaria, February 15-19, 2010. Proceedings. (2010) 286–305
11. Rogozan, D., Paquette, G.: Managing ontology changes in the semantic web. In: Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, Compiegne, France, IEEE publication (2005) 430–433
12. Zablith, F.: Evolva: A comprehensive approach to ontology evolution. In: Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications. ESWC 2009 Heraklion, Berlin, Heidelberg, Springer-Verlag (2009) 944–948
13. Sellami, Z., Camps, V., Aussenac-Gilles, N.: DYNAMO-MAS: a multi-agent system for ontology evolution from text. J. Data Semantics **2**(2-3) (2013) 145–161
14. Paulheim, H., Bizer, C.: Improving the quality of linked data using statistical distributions. International Journal Semantic Web Information System **10**(2) (2014) 63–86
15. Sören, A., Lorenz, B., Christian, D., Orri, E., Michael, H., Robert, I., Jens, L., Michael, M., Pablo, N., Bert, V.N., Stadler, C., Tramp, S., Williams, H.: Managing the life-cycle of linked data with the LOD2 stack. In: The Semantic Web - ISWC 2012 - 11th International Semantic Web Conference, Boston, MA, USA. (2012)
16. Dirschl, C., Eck, K., Lehmann, J.: Supporting the data lifecycle at a global publisher using the linked data stack. European Research Consortium in Informatics and Mathematics News **96** (2014) 38–39
17. Cabrio, E., Villata, S., Gandon, F.: Classifying inconsistencies in dbpedia language specific chapters. In: Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014. (2014) 1443–1450
18. Laera, L., Blacoe, I., Tamma, V., Payne, T., Euzenat, J., Bench-Capon, T.: Argumentation over ontology correspondences in MAS. In: Proceedings of 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS. (2007) 228–238