J Data Semant (2013) 2:145–161
DOI 10.1007/s13740-013-0025-1

ORIGINAL ARTICLE

DYNAMO-MAS: a Multi-Agent System for Ontology Evolution from Text

Zied Sellami · Valérie Camps · Nathalie Aussenac-Gilles

Received: 2 September 2012 / Revised: 27 November 2012 / Accepted: 20 December 2012 / Published online: 28 May 2013

Abstract Manual ontology development and evolution are complex and time-consuming tasks, even when textual documents are used as knowledge sources in addition to human expertise or existing ontologies. Processing natural language in text produces huge amounts of linguistic data that need to be filtered out and structured. To support both of these tasks, we have developed DYNAMO-MAS, an interactive tool based on an adaptive multi-agent system (adaptive MAS or AMAS) that builds and evolves ontologies from text. DYNAMO-MAS is a partner system to build ontologies; the ontologist interacts with the system to validate or modify its outputs. This paper presents the architecture of DYNAMO-MAS, its operating principles and its evaluation on three case studies.

Keywords Ontology engineering from text · Ontology evolution · Adaptive multi-agent system

1 Introduction

Ontologies are often used to represent a specification of domain knowledge that reflects a consensual agreement on the domain or the knowledge required for a specific application. Ontologies are now essential in many applications (access to shared knowledge, semantic web, information retrieval, etc.). An ontology model is made of concepts structured by hierarchical and non-hierarchical semantic relations.

Building and evolving an ontology relies on elaborated processes to identify the right domain knowledge and to formalize it in a relevant way, so that it could be usable by software applications and in some cases, interpretable by humans [26]. These processes can be time-consuming. They require a constant verification of the ontology with regard to the applications using it as well as the evolutions of the domain being modeled.

Indeed, modeling assumes to make various decisions that organize, classify, and describe the concepts of a specific domain according to the point of view that best suits the use of the ontology. Whatever the domain and the application, several possibilities exist to organize knowledge and it is difficult to estimate the "best" one [11].

In recent years, the use of text (as knowledge source) and language processing (as a technique for rapidly detecting linguistic clues of knowledge in text) have emerged as a solution to make this task easier [8]. Practically, text-based approaches rely on the ontologist[1] to decide on how to organize knowledge in the ontology, but they provide him with data for conceptual structuring, e.g., some candidate phrases can become concept labels, classes of phrases can contribute to define concepts, lexical relations can serve as clues for semantic relations, etc.

Critical steps for the success of ontology evolution from linguistic material include (i) the choice of a relevant knowledge-rich corpus; (ii) the selection of appropriate NLP software tools, adapted to the corpus content and to the sought knowledge; (iii) the identification of relevant elements

Z. Sellami · V. Camps (✉) · N. Aussenac-Gilles
Institut de Recherche en Informatique de Toulouse,
IRIT - UMR 5505 Université Paul Sabatier, 118 Route de
Narbonne, 31062 Toulouse Cedex 9, France
e-mail: camps@irit.fr

Z. Sellami
e-mail: sellami@irit.fr

N. Aussenac-Gilles
e-mail: aussenac@irit.fr

[1] The ontologist is the person in charge of building and evolving an ontology. Either a domain expert or an ontology expert, he knows the needs to be satisfied by the ontology and the application that will use it.

among the large amount of linguistic data; and finally (iv) an appropriate formalization of these elements [3].

We are particularly interested in the last two steps: we want to assist an ontologist who updates an ontology. First, we want to help him to identify, within the large quantity of linguistic data that can be extracted from text, those relevant for the considered field and for the target application.

Next, we want to guide him by proposing a first organization of these data within a structured model. To this end, we propose a support software tool (DYNAMO-MAS) that *co-constructs* the ontology jointly with the ontologist: (a) the system provides proposals to the ontologist (such as new concepts, new relations, new terms) who (b) validates, corrects or rejects them, and (c) the system learns from the answers that it receives to later provide better proposals. This cyclic process can be activated whenever new resources are available or when new needs show that the ontology should be modified. It is particularly fitted to the management of ontologies in dynamic environments.

To implement these principles, we rely on adaptive multi-agent systems (adaptive MAS or AMAS). Adaptive software agents are defined to represent the ontology being modified and the linguistic data extracted from the corpus. The agents' behavior is guided by parameters evaluated from the word usages in the corpus and by heuristics for structuring the ontology. To keep track of the linguistic formulations of concepts, a terminological component is added to the ontology to form a Termino-Ontological Resource (TOR). According to the TOR data model, terms are represented as classes and linked to concept classes by a *denotation* relation. This type of ontology is particularly relevant for text semantic annotation because terms contribute to link ontologies, semantic annotations, and texts.

This paper presents the design and evaluation of DYNAMO-MAS, an interactive software based on a MAS that aims at evolving ontologies from text. Section 2 offers an overview of the DYNAMO project. The architecture of DYNAMO-MAS is then detailed in Sect. 3. Section 4 contains the experiments of ontology evolution that were performed with DYNAMO-MAS and an analysis of their results. Section 5 describes related works regarding existing tools for building and evolving ontologies from text. We conclude and plan some future works in Sect. 6.

2 Scientific Context

2.1 The DYNAMO Project

Ontology evolution is the core of the DYNAMO (DYNAMic Ontology for information retrieval)[2] project.

This project addresses the improvement of semantic information retrieval driven by users' satisfaction in a dynamic context. It aims at proposing a method and a set of tools that enable the definition and the evolution of ontological resources from a set of textual documents. These resources are used to facilitate information retrieval within the corpus by means of semantic indexing.

One of the DYNAMO main originalities is to take into account the potential dynamics of the searched document collection, of the domain knowledge as well as the evolution of users' needs. The document collections are supposed to be task-oriented technical documents. They have a reasonable size (a few hundred documents) that make them manageable by a human who checks their annotation. DYNAMO does not aim at dealing with very large web-extracted corpora.

In this project, three corpora are provided by partners coming from three different domains: research in archeology about techniques, automotive diagnosis (automotive components, symptoms, engine failures, etc.), and software bug reports. The first two corpora are written in French, while the last one is written in English. These corpora serve as knowledge sources and are used to update each domain ontology. Two ontology evolution approaches are performed in DYNAMO: (i) EvOnto, that deals with the joint evolution of an ontology and semantic annotations [41] and (ii) DYNAMO-MAS, an adaptive AMAS that supports ontology evolution. DYNAMO-MAS is the subject of this article.

2.2 The Ontology Meta-model in the DYNAMO Project

In the DYNAMO project, the ontology and its lexical components form what we call a TOR. A TOR consists of a conceptual component (the ontology) and a lexical component (the terminology) [2,10,26].

In DYNAMO, a TOR (called "ontology" in the rest of the paper) is formalized using the OWL[3]-based TOR model proposed in [35]. This model recently became a meta-model, where concepts and terms are two meta-classes adapted from owl:class. In this model, concepts are related to their linguistic manifestations in documents (i.e., terms): a term "denotes" at least one concept. For example, in a TOR representing the automotive diagnosis field, the *car* concept is denoted by several terms such as *car, vehicle, and automobile*.

2.3 Why Use of a Multi-agent System?

The DYNAMO-MAS tool aims at reducing the need for the ontologist's intervention when modifying an existing ontology from textual resources. From the results of text analysis and the current state of an ontology, the MAS gener-

[2] http://www.irit.fr/dynamo/ The DYNAMO Projet benefits of the ANR (French National Research Agency) grant ANR-07-TLOG-004.

[3] OWL: web ontology language.

ates a new version of this ontology. Results are then displayed to the ontologist for validation. The ontologist can either approve, move, or reject each of the terms, concepts, relations, and denotation links proposed by the MAS. His approval or changes are sent back to the MAS that accordingly updates its knowledge. Then the MAS self-organizes and proposes a new ontology.

To define and implement this MAS, we used the Adaptive Multi-Agent System (AMAS) paradigm [9,17]. This paradigm is based on an organizational approach particularly relevant to build MAS that continuously and locally self-adapt to the dynamics of the environment. AMAS agents generally are numerous. Each of them has a local knowledge on itself and on "neighbor" agents with whom it already has interacted. Several types of agents can interact in one single system. Each type of agent behaves simply according to cooperative interaction rules. Its knowledge and its behavior depend on the application, but its rules have to guaranty that all the agents collectively achieve the intended overall complex function.

The collective function of DYNAMO-MAS is to provide an ontology that is no longer challenged by the ontologist. Two types of agents have been defined in DYNAMO-MAS : *concept* agents and *term* agents. The output TOR is built from the MAS subset whose agents evaluate themselves as good candidate terms or good candidate concepts. Each agent knows its relations with other agents, some linguistic elements from the corpus (terms and lexical relations) and manages some parameters. Each agent aims at finding its "right position" (the one that optimizes some of its parameters) in the MAS. An agent can appear, disappear, or change its relations with other agents. The MAS has to react to the perturbations generated by the addition of new documents to the corpus and/or by the ontologist's actions. To reach this goal, each agent composing the MAS and involved in these perturbations, has to self-adapt accordingly by updating its knowledge on existing agents, by creating or removing agents and relations with agents, and/or by communicating with other agents. When the MAS reaches a stable state, the resulting organization defines a new version of the ontology. The architecture and the DYNAMO-MAS principles are detailed in Sect. 3.3.

The use of an adaptive MAS is motivated by the following reasons:

- The MAS runs identically when building a new ontology or when evolving an existing one (building can be seen as a subclass of evolution where the initial ontology is an empty one). Ontology building and evolution are incremental processes: when new data is added to the corpus, the ontology evolves without resuming the evolving process from scratch.
- The linguistic data extracted from text have properties that are suited to define agents: terms are numerous; they have a local "knowledge" on the neighbor terms that they are related with in a given pattern; and they may (or not) be good candidates for the ontology.
- The practices of ontology building from text enable to define some simple rules, heuristics, or numerical parameters (such as frequency or productivity of a term) that lead to validate or reject a term, to move a term closer to other terms (that have a similar linguistic behavior or similar neighbor terms in corpus for example), to decide where to set a concept in the ontology (differentiation criteria or common properties), etc. Various heuristics have been adapted to implement the behavior of each type of agent and the way they react to environmental perturbations (the ontologist's actions or the addition of documents to the corpus).

From a practical point of view, an agent behavior defines how to react to pieces of information received from other agents and how to communicate with them. Agent parameters and knowledge have been defined to account for the relevance of linguistic elements with regard to the ontology under construction.

3 The DYNAMO System

A first version of DYNAMO-MAS has been presented with a single and basic experimentation in [38], [37], and [36]. Since then, the agents' behavior has been improved and new experimentations have been carried out. This paper provides an overall view on the evolution process, details a more efficient agent behavior, and provides promising results from new experimentations.

3.1 The Evolution Process and the Architecture of the System

The evolution process in DYNAMO is based on extracting terms and lexical relations from text, on providing such data to agents that react and self-organize, and finally on translating some of the agents into an OTR representation. The OTR on the right of Fig. 1 is relevant to the text on the left. Underlined words in the corpus correspond to a linguistic manifestation of a lexical relation between terms (terms marked in bold).

When new documents are added to the corpus (Fig. 2), new knowledge can appear. If the OTR is supposed to annotate the document on the left of Fig. 2, it needs to contain new terms and their corresponding concepts: *automobiles, airplanes*, *aircrafts*, and *means of air transport*. One can rely on the lexical relations in the text to decide how to integrate them

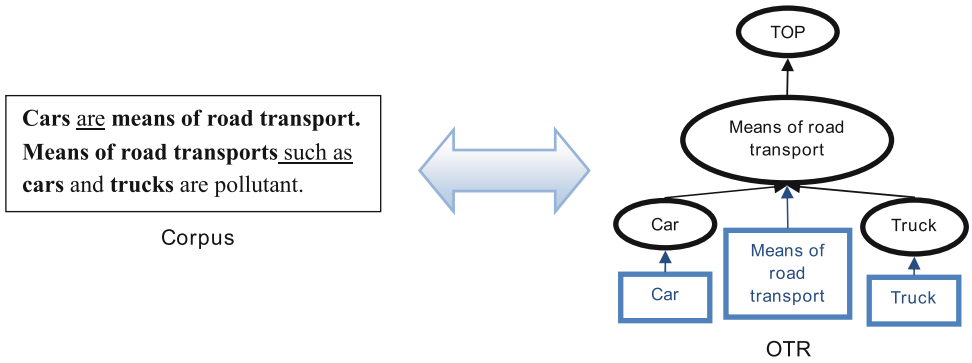

Fig. 1 Part of an OTR about means of transport [ellipses (resp. rectangles) represent concepts (resp. terms)]

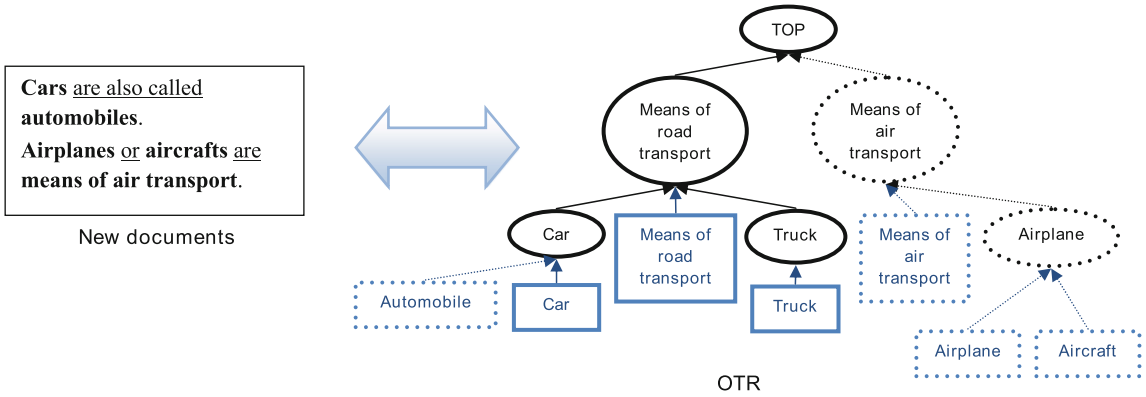

Fig. 2 Evolution of the OTR about means of transport after adding a new document

in the ontology. For instance, the expression *are also called* reveals that the *automobiles* and *cars* terms are similar or synonymous. So the ontologist can add *automobile* as a new term denoting the *Car* concept.

The overall evolution process that includes the use of DYNAMO-MAS can be decomposed into the following four steps:

1. Adding new documents to a corpus;
2. Extracting and selecting new terms and lexical relations from the new documents (and only those);
3. Semantic interpretation of the selected terms and lexical relations to update the ontology;
4. Evaluation of the ontology evolution suggestions: the ontologist controls the quality of the system proposals and makes additional modifications.

The architecture of DYNAMO (Fig. 3) is composed of three modules: steps 2 and 3 are implemented respectively in DYNAMO Corpus Analyzer and DYNAMO-MAS, whereas step 4 is supported by the "proposal manager" module. Each module is detailed in the following sections.

3.2 The Corpus Analyzer

The goal of the corpus analyzer is to prepare inputs for the MAS. It runs the YaTeA [1] *term extractor*, a *lexical relation*

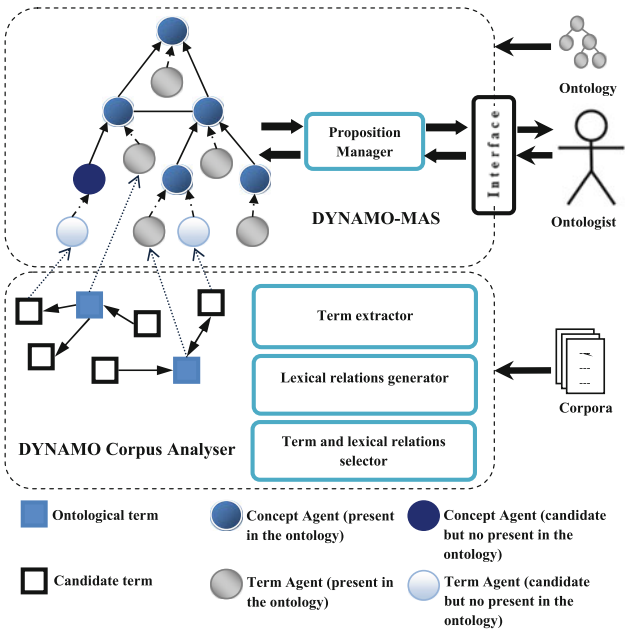

Fig. 3 Architecture of the DYNAMO system

generator, and a *term and lexical relations selector*. It outputs $< T_i$, Rel, $T_j >$ triplets where T_i and T_j are candidate terms and/or terms (if the term belongs to the ontology) and Rel is a lexical relation label. Each triplet has a confidence score (Q, I) where Q is the quality of the relation and I is the number

of occurrences of this relation in the corpus. Four types of lexical relations are generated by the relation extractor. The type of a relation determines the ontological knowledge that the MAS will define from a triplet:

1. *Hyperonymy*: it expresses a generic-specific relation between terms and may lead to define *is_a* relation between the concepts denoted by these terms;
2. *Meronymy*: it expresses a part-hood relation between terms and may lead to define a *part_of* or an *ingredient_of* or a *member_of* relation between the concepts denoted by these terms;
3. *Synonymy*: it may lead to connect the related terms to the same concept with a *denote* relation;
4. *Other relations* (called "transverse relations") are domain specific. They may lead to define specific semantic relations between the concepts denoted by the related terms, such as *causes*, *contributes_to*, *affects*, etc..

Lexical relations can be extracted by three means: (i) by matching lexico-syntactic patterns, the four kinds of relations can be found; (ii) syntactic term inclusions extract hypernymy relations; and (iii) similarity between (candidate) terms (measured with the Levenshtein distance [24]) is used to compute synonymy relations.

Several lexico-syntactic patterns have been defined to extract each type of relation. In a specific domain, a pattern conveys a regularity in the linguistic expression of a lexical relation. We assume that all the term pairs found by matching the same pattern are linked with the same type of relation. However, a pattern can extract an invalid relation. So we have defined a pattern confidence score composed of Q (quality of the extraction) and I (number of pattern occurrences). The Q quality of the extraction is a value ranging from 1 to 10, and given by the person who defines the pattern from her empirical evaluation of the pattern occurrences. The correctness of the extracted relation is directly correlated with the pattern quality value. In other words, a low quality value for a pattern means that any relation extracted with this pattern is likely to be wrong.

In order to select the candidate terms for which an agent will be defined (in the text that follows, we will call this process to *agentify* a term) and the lexical relations to be processed, we defined two confidence scores: a *confidence score for relations* and a *confidence score for terms*. Like the confidence score defined for patterns, a confidence score consists of a pair (Q, I).

The confidence score of relation R_i between terms[4] T_x and T_y is the pair (Q_{R_i}, I_{R_i}) such that:

- Q_{R_i} is the maximum quality of the relation. It is equal to the maximum quality of a lexico-syntactic pattern that matches R_i relation;
- I_{R_i} is the sum of the instances of the lexico-syntactic patterns having a quality Q_{ia} with a distance to Q_{R_i} lower or equal than δ_Q.

The (Q, I) confidence score of a relation is normalized according to the following formula:

$$Conf\left(R_i(T_x, T_y)\right) = (Q_{R_i}, I_{R_i}) = \left(Max(Q_{ij}); \sum_{a=1}^{n} I_{ia} | Q_{ia} \geq Max\, Q_{ij}) - \delta_Q\right)$$

- T_x and T_y are two terms;
- R_i is a lexical relation i between terms T_x and T_y;
- Q_{ij} (resp. Q_{ia}) is the quality of the lexico-syntactic pattern j (resp. a) for relation i;
- I_{ia} is the number of occurrences of lexico-syntatic pattern a for relation i;
- δ_Q is a threshold that determines the patterns taken into account when counting the relation instances. Only patterns with a quality score Q_{ia} close to the best one are selected. We arbitrarily set δ_Q to 0.5 (5 % of the maximum value of Q which is 10) . If δ_Q is set to 0, only instances of the best pattern will be counted. If δ_Q equals 10 (maximum value), instances of all available patterns for relation R_i will be taken into account.

The Q_{R_i} confidence score of a relation increases with the quality of the patterns that identify it. The number of occurrences of this relation (I_{R_i}) is the sum of the number of occurrences of the patterns with a quality score close to the maximum value (more or less). I increases when Q decreases: the lower the quality threshold, the more patterns and relation occurrences are taken into account, even those who have no meaning. This is why we set constant δ_Q to 0.5: we want the system to consider relations of good quality [(Q is close to the Maximum Quality (QR)], but not only the best one, also those with a quality "close to" the best one. Because Q1 and Q2 values range between 0 and 10, with δ_Q equal to 0.5, "close to" means a 5 % difference.

When a relation is found not using pattern matching, but thanks to one of the other implemented methods (syntactic analysis and the Levenshtein distance), I is set to 1; and the quality Q is set to the confidence score of the results provided by these tools.

The confidence score of term T_x is represented by the pair (Q_{T_x}, I_{T_x}) such that:

- Q_{T_x} is the maximum quality of all the relations involving T_x;

[4] either belonging to the TOR or being candidate terms.

– I_{T_x} is the sum of all the occurrences of the relations involving T_x, the quality Q of which has a distance smaller than δ_Q to the maximal quality Q_{T_x}.

A good candidate term is one that is involved in high quality relations. So the confidence score of relations is used in the evaluation of the term confidence score. Moreover, the formula to estimate the confidence score of term T_x is similar to the one defined for the relation confidence score:

$$Conf(T_x) = (Q_{T_x}, I_{T_x}) = \left(Max(Q_{R_i}); \sum_{a=1}^{n} I_{R_a} | Q_{R_a} \geq Max(Q_{R_i}) - \delta_Q \right)$$

– Q_{R_i} is the quality of relation R_i involving T_x;
– I_{R_a} is the number of instances of relation R_a involving T_x;
– Q_{R_a} is the quality of relation R_a involving T_x;
– δ_Q is arbitrarily set to 0.5 to select the relations with the quality close to the best one.

These two formulas are used to select the most relevant candidate terms and relations. We chose to define a *term* agent from each candidate term with a quality score above the average (5). All the candidate terms that have a score lower than the average are then considered to be noise. If all candidate terms have a quality score below average, no proposal is made to the ontologist. In this case, we consider the average of all the qualities of all candidate terms as a selection criterion.

3.3 DYNAMO-MAS: the Multi-Agent System

The MAS includes the ontology that will be proposed to the ontologist. Its inputs are the triplets provided by the corpus analyzer and an OWL ontology. It provides as output an ontology expressed according to the OWL meta-model presented in Sect. 2.2.

The MAS consists of two types of agents: (i) *term* agents represent candidates for the terminological part of the ontology and (ii) the *concept* agents represent candidates for the conceptual part of the ontology. The definition of two types of agents partly reflects the structure of a DYNAMO TOR: a concept (resp. term) of a DYNAMO ontology corresponds to a *concept* agent (resp. a *term* agent). The relations of the ontology as well as the linguistic relations are part of an agent knowledge and are not agentified. This option is motivated by the need to reduce the number of agents and the computation time of the system, and also to restrict and to simplify the interactions between agents. Thus, relations between concepts (resp. terms) are represented as links between *concept* (resp. *term*) agents in each of the related *concept* (resp. *term*) agent, and denotation relations between terms and concepts are links between *term* agents and *concept* agents. All the links are part of the knowledge of the concerned (*term* or *concept*) agent.

The algorithms (or behaviors) of the two types of agents composing the MAS are presented in this section. The initial state of the MAS is an organization composed of *concept* agents and *term* agents created by "agentifying" concepts and terms of an existing ontology. Each agent knows its confidence score (Q, I). Each *concept* agent knows the *concept* agents to which it is related and the relation name, as well as the *term* agents that denote it. Each *term* agent knows the *concept* agents that it denotes, the *term* agents with which it is lexically related, and the name of these relations.

Currently, DYNAMO-MAS only suggests enriching the initial ontology with new concepts, terms, or relations without any change on the original content; if needed, it is up to the ontologist to manually modify these initial elements. This solution has been adopted at the request of the project partners. However, DYNAMO-MAS is able to suggest changes to concepts, terms, and relations that were in the ontology before its evolution (by adding an intermediate concept between two concepts, by changing relations between concepts and terms, etc.). To make these changes possible, one just needs to activate those features in the agents.

3.3.1 Term Agents

The *term* agents represent the terminological part of an ontology. A *term* agent has a status (*candidate term* or *term*) indicating whether the agent is actually part of the ontology or is at proposal stage. Each agent is connected to other *term* agents in accordance with the lexical relations extracted from the corpus. According to the meta-model of the ontology (that requires that each class term is connected by at least one denotation relation to a class concept), a *term* agent must be connected to at least one concept. Each relation between two *term* agents $< T_i$, Rel, $T_j >$ is assigned a confidence value (Q, I).

A *term* agent has to achieve three objectives: (i) *to denote at least one* concept *agent*, (ii) *to process all its lexical relations*, and (iii) if possible, *to propose itself to the ontologist*. During its life cycle, the *term* agent processes its goals as well as the requests received from other agents (from the highest priority to the lowest priority). Its primary objective is to denote a *concept* agent. Other priorities are determined by the agent according to the confidence in the relation to be processed, the relevance of the agent for proposing itself to the ontologist, and the priorities of the requests it receives. The general algorithm of a *term* agent is Algorithm 1.

Algorithm 1: General algorithm of a *term* agent

```
1  Perceive();
2  // It perceives the messages sent by agents and by the
3  // corpus analyzer ;
4  ProcessNotification();
5  // It updates its knowledge on its relations  ;
6  // (acceptation, reject and suppression of relations) ;
7  if (!havingADenotationRelation and !isInOTR) then
8      ProcessNoDenotationLinkProblem();
9      // It processes the denotation relation;
10 end
11 Task mostCriticalTask ⟵ getMostCriticalTask();
12 if (mostCriticalTask != null) then
13     if (mostCriticalTask instanceof IRequest) then
14         ProcessRequest(mostCriticalTask);
15         // It processes its requests from the most
16         // critical one to the least critical one;
17     else
18         ProcessRelation(mostCriticalTask);
19         // It processes relations from the most critical
20         // one to the least critical one;
21     end
22 else
23     if (!isInOTR) then
24         float agRelevance ⟵ updateAgRelevance();
25         if (agRelevance ≥ termAgRelevance) then
26             ProposeToTheOntologist();
27             // If its confidence is higher than the
28             // threshold, it proposes itself to be part
29             // of the ontology;
30         end
31     end
32     stopAgent();
33 end
```

(i) In order to *denote a* ***concept*** *agent*, a *term* agent asks for the creation of a *concept* agent (Fig. 4). A *concept* agent is created if a *concept* agent having the same label as the *term* agent does not exist yet in the MAS. Then the *term* agent sends back to the *concept* agent a denotation request (❶). This request is always accepted by a *concept* agent (❷). This denotation link has a confidence value calculated by the *term* agent according to the formula given in Sect. 3.2.

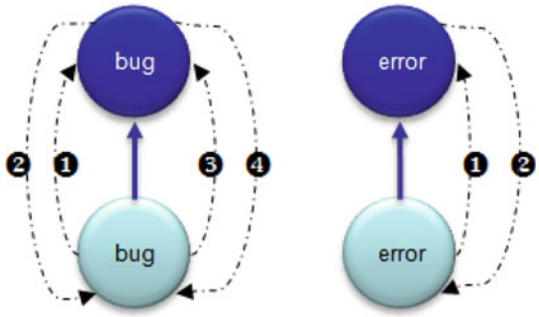

Fig. 4 Interactions between *term* agents and *concept* agents

(ii) A *term* agent *processes its outgoing lexical relations*. Every relation has a confidence value and a status (unprocessed, processed, or rejected). An agent considers its relations from the most relevant one (with the highest confidence value) to the less relevant one by sending to the *concept* agents; it denotes a request to process the lexical relation (❸). The *concept* agent processes the relation and then it notifies the *term* agent with a message of acceptance or reject with a (Q, I) confidence score (❹). The *term* agent updates the status of the relation that has been processed.

Let's assume that the `bug` *term* agent processes a synonymy relation between the terms `bug` and `error` (figure 5, ❺). The `bug` *concept* agent that it denotes sends to `error`-term a request to change its denotation link (❻). If the (Q, I) score of the current denotation link of `error`-term is lower than the value of the request, it accepts the request, changes its denotation link (❼) and notifies `bug`-concept of its agreement (❽). Otherwise, it notifies `bug`-concept of its reject with the confidence score of the denotation link between `error`-term and `error`-concept. `bug`-term (that sent the first request) is then notified (❾).

The status of a relation may change at each iteration of the algorithm after the addition of new documents to the corpus. The *term* agent may request again the

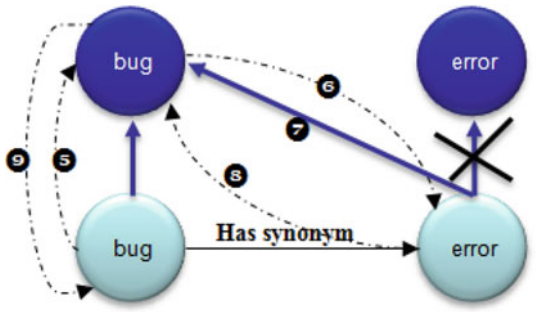

Fig. 5 Interactions between *term* agents and *concept* agents to manage a synonymy relation

processing of a *rejected* relation if its confidence value increases and exceeds the rejection threshold.

(iii) In order to *propose itself to be part of the ontology*, a *term* agent computes its relevance score according to the following formula:

$$termAgRelevance = \alpha_1 * P_1 + \alpha_2 * P_2 + \alpha_3 * P_3 + \alpha_4 * P_4$$

where P_1 is the maximum value of all its lexical relations; P_2 is the accuracy of its *term* agent neighbors; P_3 expresses the accuracy of the *term* agent lexical relations; P_4 expresses the diversity of the *term* agent lexical relations and $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ are the different weights of the P_i. More precisely:

- P_1 = MaxLexicalRelationConfidence = $Max(Q_{R_i})$;
- P_2 = (nbRTAInTOR - nbRTANotInTOR)/ (nbRTAInTOR + nbRTANotInTOR) where RTA (Related Term Agent) are *term* agents in relation with the evaluated *term* agent;
- P_3 = (nbAcceptedLRA - nbRefusedLRA) / (nbAcceptedLRA + nbRefusedLRA) where LRA (Lexical Relation of the Agent) are lexical relations known by the evaluated *term*;
- P_4 = nbDifferentAgentLexicalRelation / nbAllPossibleLexicalRelation.

After various experiments with DYNAMO-MAS, we empirically fixed the values of α_1 to 0.5, α_2 to 2, α_3 to 2 and α_4 to 1. These values best weighed the parameters of the agent relevance.

The relevance of an agent is the maximum confidence score of all its relations with its neighboring agents (*concept* agents and *term* agents). Above a threshold (between 1 and 10), a *term* agent proposes itself to be part of the ontology by sending a request to the proposal manager. The user can tune this threshold starting with a low value at the beginning of the evolution process (to get a lot of suggestions) and increasing this value when the ontology provides good annotations for all the documents in the corpus.

When the ontologist accepts or rejects a proposal, the proposal manager notifies the *term* agent by a reject or an acceptance. The *term* agent updates its status. To prevent *term* agents to propose themselves again after the ontologist has rejected them, a high value (8) has been assigned to the ontologist's interventions (this parameter can be adjusted according to the application). Indeed, during the ontology evolution, an ontologist can qualify a candidate term as irrelevant for the domain and can eventually later reverse his decision. In this case, the *term* agent can propose itself again if its relevance value exceeds the ontologist's reject value.

3.3.2 Concept *Agents*

Concept agents represent the conceptual part of an ontology. A *concept* agent has a status (*candidate concept* or *concept*) indicating whether the agent is part of the ontology or is at proposal stage. A *concept* agent is connected to other *concept* agents by conceptual relations. It is also connected to *term* agents by denotation links. Its relations can have the status *unprocessed*, *processed*, or *rejected*. A *concept* agent has to achieve its three objectives: (i) *to have semantic relations with* concept *agents and denotation links with* term *agents*; (ii) *to determine a preferred label*, and (iii) *to propose itself to the ontologist*. As for *term* agents, the *concept* agent processes its objectives from the highest priority to the lowest priority. The general algorithm of a *concept* agent is Algorithm 2.

(i) A *concept* agent receives requests from other *term* agents (Fig. 4, ❶) to process lexical relations and requests from *concept* agents to establish conceptual relations (❷). When a *concept* agent processes a request from a *term* agent to *process a lexical relation*, it sends a request to the *concept* agents denoted by the second *term* agent in the lexical relation (❷). It asks these *concept* agents to establish a conceptual relation with itself. For example (Fig. 6), when `applet`-concept is inforbreakmed that `applet`-term has `program`-term as hypernym, `applet`-concept asks `program`-concept to establish the "`applet`-concept *is_a* `program`-concept" relation. A *concept* agent can either accept or refuse to take into account a relation with another *concept* agent. It rejects it if a stronger conceptual relation already exists between itself and the other agent. Here, `program`-concept sends back a positive notification of its decision to `applet`-concept (❸). But, if a part-of relation between `applet`-concept and `program`-concept was requested by `applet`-concept to `program`-concept, then `program`-concept compares the confidence of these relations and only keeps the relation with the highest score. When receiving a notification, `applet`-concept updates the status of the concerned relation and updates its links with `program`-concept (❹). `applet`-term is then notified by `applet`-concept that has established the conceptual relation (❺).

(ii) To *determine a preferred label*, the *concept* agent choses the name of the *term* agent having the denotation link with the highest weight. This label can evolve if new *term* agents denote the *concept* agent or if the value of the denotation links changes.

(iii) In order to *propose itself to the ontologist*, in a similar way as a *term* agent does it, a *concept* agent checks its relevance value. If it is higher than the relevance thresh-

old, it proposes itself. The relevance value is evaluated according to the following formula:

$$conceptAgRelevance = \alpha_1 * P_1 + \alpha_2 * P_2 + \alpha_3 * P_3 + \alpha_4 * P_4 + \alpha_5 * P_5$$

Algorithm 2: Main algorithm of a *concept* agent

```
1  Perceive();
2  // It perceives the messages sent by other agents;
3  ProcessNotification();
4  // It updates its knowledge on its relations ;
5  // (acceptation, reject and suppression of relations) ;
6  if (((havingADenotationRelation and !havingALabel)
   or denotationRelationUpdated) and !isInOTR) then
7      DetermineALabel();
8      // It processes its preferred label if it is given a ;
9      // new one denotation link or if the confidence ;
10     // values in its relations are updated;
11 end
12 Task mostCriticalTask ⟵ getMostCriticalTask();
13 if (mostCriticalTask != null) then
14     ProcessRequest(mostCriticalTask);
15     // It processes its requests from the most critical ;
16     // one to the least critical one ;
17 else
18     if (!isInOTR) then
19         if (uselessAgent) then
20             Disappear();
21             // It is useless, it disappears;
22         else
23             float agRelevance ⟵
               updateAgRelevance();
24             if (agRelevance ≥ conceptAgRelevance)
               then
25                 ProposeToTheOntologist();
26                 // If it exceeds the relevance threshold,
27                 // it proposes itself to the ontologist ;
28             end
29         end
30     end
31     stopAgent();
32 end
```

where P_1 is the maximum confidence value of all its conceptual relations; P_2 is the accuracy of the *concept* agents that are in relation with; P_3 expresses the accuracy of the conceptual relations of the *concept* agent ; P_4 is the depth in the ontology; P_5 is the proportion of relevant *term* agents that denote this *concept* agent and $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5$ are the different weights of P_i. More precisely:

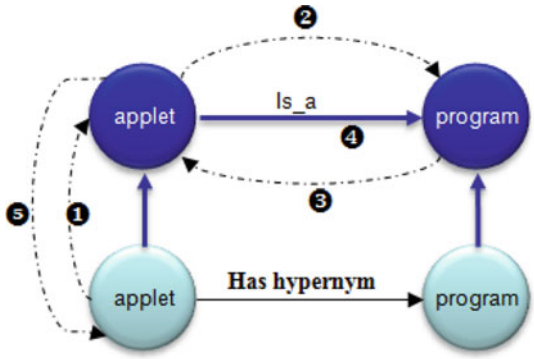

Fig. 6 Interactions between *term* agent and *concept* agent to establish an *is_a* relation

- P_1 = MaxConceptualRelationConfidence;
- P_2 = (nbRCAInTOR - nbRCANotInTOR) / (nbRCAInTOR + nbRCANotInTOR) where RCA (Related Concept Agent) are *concept* agents in relation with the evaluated *concept* agent;
- P_3 = (nbAcceptedCRA - nbRejectedCRA) / (nbAcceptedCRA + nbRejectedCRA) where CRA (Conceptual Relations of the Agent) are conceptual relations known by the evaluated *concept*;
- P_4 = {-1;1} : -1 if the *concept* agent is connected to the top agent of the ontology, 1 otherwise;
- P_5 = (nbRelevantTermAgent - nbNotRelevantTermAgent) / (nbRelevantTermAgent + nbNotRelevantTermAgent) where the *term* agents are denoting the *concept* agent.

After some experiments with DYNAMO-MAS, we empirically fixed the values of α_1 to 0.5, α_2 to 1, α_3 to 1, α_4 to 1, and α_5 to 2.

The processing of synonymy relations implies changing the denotation link between *term* agents and *concept* agents. If a *concept* agent is no more connected to any agent, it cannot receive any request and cannot reach any of its objectives: it becomes useless in the MAS and disappears.

3.3.3 The Proposal Manager

The proposal manager enables the ontologist to both visualize the ontology and the MAS proposals and to interact with the MAS (Fig. 3). Its main goal are (i) to sort out the proposals sent by the *concept* agents and the *term* agents for display *via* the GUI and (ii) to convey the decisions made by the

ontologist (acceptance, reject, or modification) to the *concept* and *term* agents with regard to their proposals.

The proposal manager is able to process proposals coming from *term* and *concept* agents. It stores them as long as the agents evaluate themselves as relevant for the ontology and send it some messages. Once the activity of the MAS is stabilized, i.e., when all the agents have processed the requests that they received, the proposal manager sorts out the proposals, deletes contradictory ones, and conveys the final list to the ontologist. Contradictions may appear either when an agent proposes itself but later disappears from the MAS, or when an agent proposes a conceptual relation and later proposes another relation involving the same concepts with a different relation. The system considers that the most recent one is the only one which is valid.

The proposal manager finally brings back to the agents that made the proposal the evaluation from the ontologist. This notification is a new request sent to the agents that will process it.

4 Evaluation of DYNAMO-MAS

We report the experiments that we carried out with DYNAMO-MAS and we analyze their results. DYNAMO is implemented as a plug-in in the ontology editor Protégé[5] (Fig. 7). DYNAMO-MAS extends TextViz [35], another Protégé plug-in dedicated to semantic annotation of text documents.

The agents in DYNAMO-MAS have been implemented with the *MAY: Make Yourself Agents* [6] platform [30]. *MAY* is a generator of agent APIs (*application programming interface*) written in Java. It is an Eclipse plug-in[7]. The user of *MAY* first specifies an abstract agent model using an architecture description language called muADL. The model is composed of a set of micro-components dedicated to the creation of new agents, to the communication between agents, to the management of the agent knowledge,... Then, the model and the behavior of the agents are implemented in Java classes. Interaction between agents is done via asynchronous message exchange. These messages have been defined as a Java class for the application.

When the ontologist adds new texts to the corpus, the DYNAMO-MAS is triggered. The corpus analyzer extracts new candidate terms and new lexical relations. It sends them as inputs to the MAS; new term agents are created and interact with the other agents. So, most of the agents in the MAS update their knowledge, react, and communicate with each other until a stable state is reached. The agents of DYNAMO-MAS use only a local knowledge and behavior to evaluate their relations with other agents, which is expected to set them at their best place in the organization. Then, the MAS proposes a new ontology (the initial ontology that has been enriched and modified) to the ontologist in the GUI (Fig. 7).

Changes are displayed in the concept panel (❶) and in the term panel (❷): proposed concepts and terms are the underlined one. A second Tab Widgets, called *DYNAMO - Virtual Ontologist Proposals* (❸), has been added to Protégé. It is a tabular view of the MAS proposals, incorporating non-hierarchical relations that cannot be seen in the two first panels. The ontologist validates, deletes, or modifies the concepts, terms, and relations proposed by DYNAMO-MAS *via* the Graphical User Interface (❹).

4.1 Experiment

We have carried out tests using DYNAMO-MAS with three different ontologies and associated corpora. Each ontology has been manually designed and validated with regard to the corpus annotation: it enables an annotation of its associated corpus that is precise enough to support an efficient document retrieval. In these three domains, the corpus contains a reduced number of small size documents (a few paragraphs).

- Artal ontology: an English ontology on software bug reports, made up of 887 terms and 582 concepts; the corpus is composed of 287 documents (bug report files);
- Arkeotek ontology: a French ontology on archeology about traditional techniques, made up of 733 terms and 380 concepts; the corpus is composed of 299 documents (rule-based formulation of scientific papers);
- Actia ontology: a French ontology on automotive diagnosis, made up of 579 terms and 330 concepts; the corpus is composed of 710 documents (files that report fault descriptions and repair procedures).

We have defined two scenarios to evaluate DYNAMO-MAS:

1. The first evaluation (quality evaluation) illustrates an ontology evolution process and evaluates the relevance of the DYNAMO-MAS suggestions: we check if the suggestions are considered as valid by the ontologist.
2. The second evaluation (performance evaluation) is designed to test the performance of the MAS in terms of scalability and perturbation diffusion inside the system.

[5] Protégé: http://protege.stanford.edu/.

[6] MAY: http://www.irit.fr/MAY.

[7] Eclipse: http://www.eclipse.org.

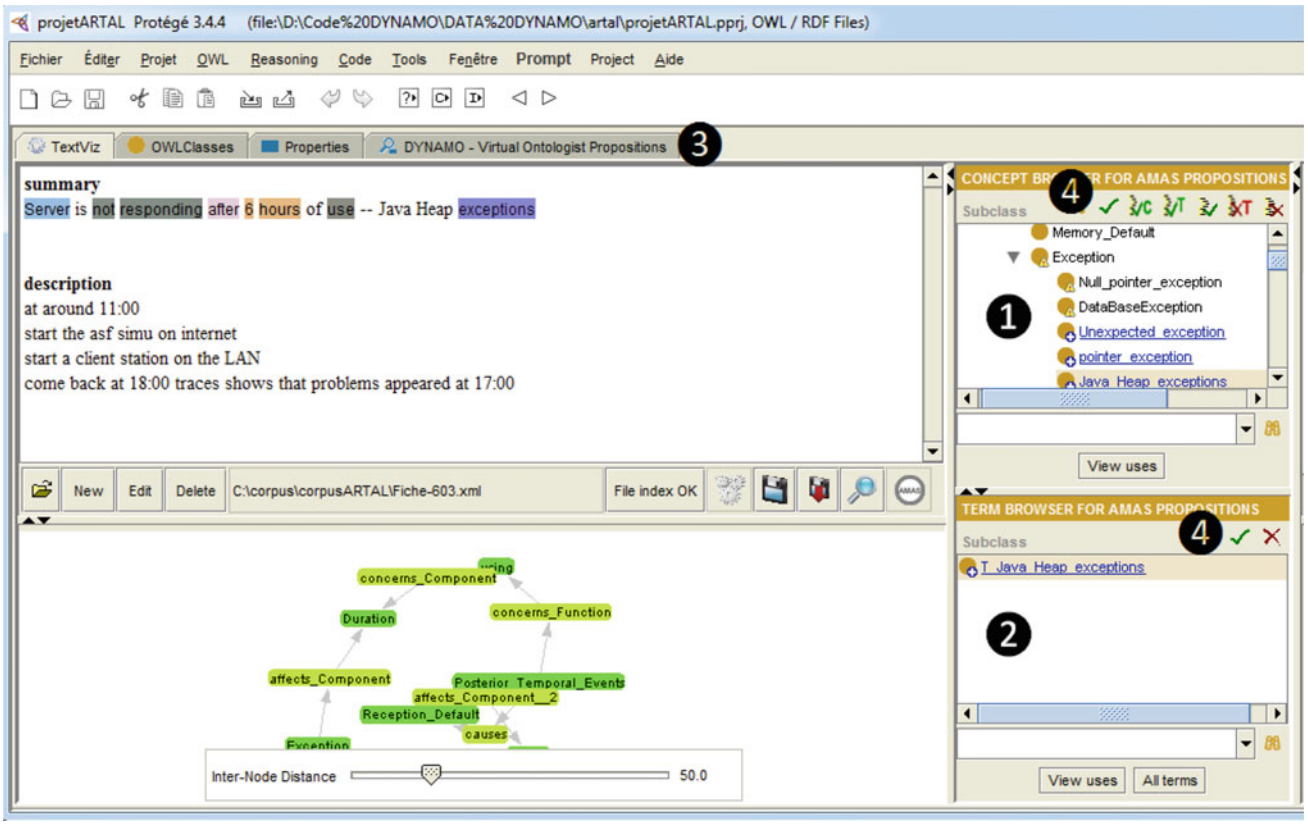

Fig. 7 The DYNAMO-MAS tool in the Protégé ontology editor

4.1.1 Quality Evaluation

To evaluate *the quality* of DYNAMO-MAS proposals, we compared a manual and an automatic ontology evolution. In each domain, an ontologist has accepted, rejected, and/or modified the MAS proposals via the graphical interface (Fig. 7).

After the addition of 21 new documents to the *Artal corpus*, the ontologist enriched the ontology with 19 new terms and 9 new concepts. Starting with the same ontology and new documents, DYNAMO-MAS proposed 24 new terms (of which the ontologist accepted 16) and 18 new concepts (of which the ontologist accepted 10). Table 1 shows a detailed report of this result.

In the Artal case-study, although DYNAMO-MAS comprises a large number of agents (1,469 agents), it is able to reach good quality results: *67 %* of term proposals are relevant and *56 %* of concept proposals are relevant. DYNAMO-MAS is also able to suggest terms and concepts that have not been identified manually (12 terms and 9 concepts). So the MAS can be seen as an interesting tool to help an ontologist to make a precise evolution of his ontology.

After the addition of 12 new documents to the *Arkeotek corpus*, the ontologist enriched the ontology with inserting 19 new terms and 7 new concepts. From the same ontology and new documents, DYNAMO-MAS proposed 32 new terms (of which the ontologist accepted 22) and 27 new concepts (of which the ontologist accepted 16). Table 2 shows the details of this result.

With the Arkeotek dataset, DYNAMO-MAS attained *68.75%* of relevant term proposals and *59.26 %* of relevant concept proposals. DYNAMO-MAS suggested 18 terms and 14 concepts not identified by the ontologist.

After the addition of 50 new documents to the *Actia corpus*, the ontologist enriched the ontology with 19 new terms and 9 new concepts. DYNAMO-MAS proposed 54 new terms (of which the ontologist accepted 9) and 30 new concepts (of which the ontologist accepted 6). Table 3 shows the details of this result.

With the Actia dataset, DYNAMO-MAS reaches only *16.98 %* of relevant term proposals and *22.22 %* of relevant concept proposals. DYNAMO-MAS has been able to suggest terms and concepts not identified manually (5 terms and 5 concepts). It obtained a poorer result with the Actia dataset because the new documents added to the corpora contain very little new knowledge. The Actia ontology covers most of the knowledge expressed in the corpus. In return, however, the Artal and Arkeotek ontologies are not achieved. In both cases, DYNAMO-MAS provides significant and helpful rewards to the ontologist.

Table 1 Results of the DYNAMO-MAS after adding 21 documents to the Artal corpus

Quality of the proposals	*Term*	*Concept*
Relevant proposals (set at the right place in the ontology)	13	5
Correct proposals (set at a wrong place in the ontology)	3	5
Acceptable proposals (useful for the domain but not accepted by the ontologist)	0	0
Useless proposals (useful for the domain but useless for the annotation process)	1	0
Wrong and rejected proposals	7	8
Proposals made both by the ontologist and the MAS	4	1

Table 2 Results of the DYNAMO-MAS after adding 12 documents to the Arkeotek corpus

Quality of the propositions	*Term*	*Concept*
Relevant proposals (set at the right place in the ontology)	18	5
Correct proposals (set at the wrong place in the ontology)	4	11
Acceptable proposals (useful for the domain but not accepted by the ontologist)	0	0
Useless proposals (useful for the domain but useless for the annotation process)	5	2
Wrong and refused proposals	5	9
Proposals made both by the ontologist and the MAS	4	2

Table 3 Results of the DYNAMO-MAS after adding 50 new documents to the Actia corpus

Quality of the proposals	*Term*	*Concept*
Relevant proposals (set at the right place in the ontology)	6	2
Correct proposals (set at the wrong place in the ontology)	3	4
Acceptable proposals (useful for the domain but not accepted by the ontologist)	1	0
Useless proposals (useful for the domain but useless for the annotation process)	18	0
Wrong and rejected proposals	27	24
Proposals found both by the ontologist and the MAS	4	1

4.1.2 Performance Evaluation

The performance evaluation studies the time performance of the MAS and it scalability. In DYNAMO-MAS, the ontology is a subset of a MAS. So, when new documents are added to the corpus, new agents are defined in the MAS that is consequently disturbed. Then, the goal of the MAS is to react to this disturbance and to come back to a stable state. To evaluate the DYNAMO-MAS *behavior*, we studied the ontology evolution when varying the number of documents added to the corpus (4, 8, 16, 32, 64, 128, 256 and 512). We measured the time required for the MAS stabilization and the number of agents in the initial MAS impacted by the new agents resulting from evolution process.

Figure 8 presents the time required for the MAS to stabilize after the addition of new agents. In the three curves, the X^2 coefficient being close to zero, the time required for the MAS to stabilize is almost linear. For example, in Fig. 8, the addition of 512 documents to the Arkeotek corpus has generated the creation of 1,468 new agents in the Arkeotek MAS (before evolution, the MAS was composed of 2,796 agents) and the stabilization time has been around 4 min and 30 s. The stabilization time of the MAS is much shorter (around 20 s) for the Artal ontology and the Actia ontology where the number of added agents did not exceed 700 agents. Figure 8 shows that DYNAMO-MAS stabilizes very quickly.

Figure 9 presents the number of affected agents after the introduction of new agents. In the three curves, we can notice that the addition of new agents to the MAS does not disrupt the whole system. For instance, when we added 512 documents to the Arkeotek corpus, the 1,468 new agents impact and change the knowledge of only 755 agents. We consider the disturbance as local.

4.2 Analysis

To sum up, DYNAMO-MAS runs ontology evolution in different domains and it supports two languages (English and French).

Despite the high percentage of bad proposals obtained in the Actia experiment, results are encouraging and the MAS approach seems relevant. In fact, even if DYNAMO-MAS is composed of more than 1,000 agents, the new *concept* agents and *term* agents use local and distributed mechanisms to set

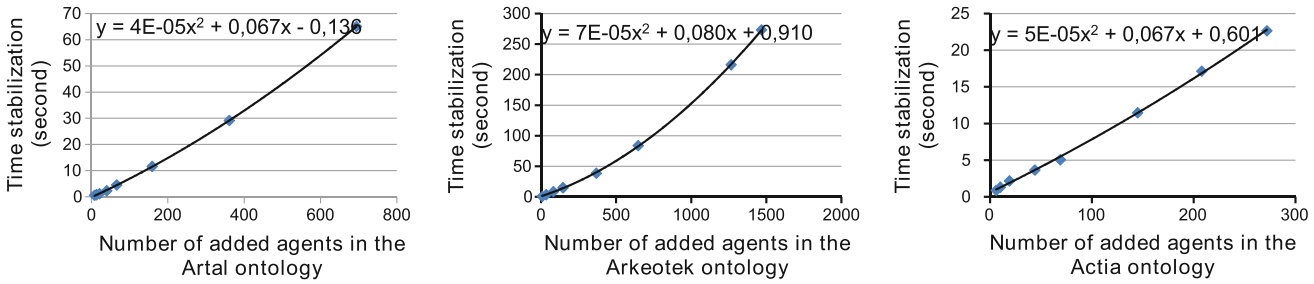

Fig. 8 Stabilization time of the MAS and number of added agents compared with the number of added documents

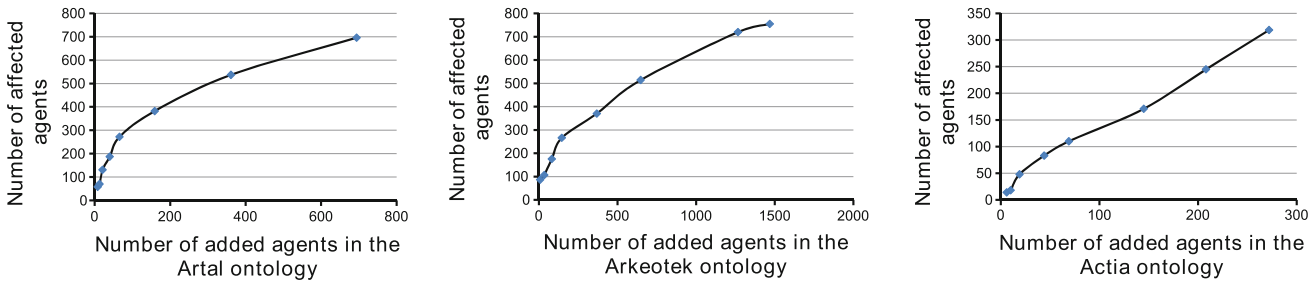

Fig. 9 Propagation of the disturbance inside the MAS

themselves at the right place in the MAS organization (the ontology). Furthermore, the MAS is able to propose many of the concepts and terms that have been added manually by the ontologist. It is also able to propose concepts and terms that have been forgotten by the ontologist.

The results provided by DYNAMO-MAS have a different quality depending on the corpus and on the initial state of the ontology. It seems that as the ontology construction progresses, the insertion of results supplied by DYNAMO-MAS becomes more complex, or conversely when the conceptualization is in its early stages, the MAS brings more help to the ontologist.

In DYNAMO-MAS, several parameters (δ_Q and α_i) were set arbitrarily and empirically after testing the system in the three experimentations. Here, we comment why and how their value could be more precisely tuned.

The δ_Q is involved in the evaluation of the quality of the Dynamo Corpus Analyzer (DCA) and in the selection of the concepts and relations that will be agentified. So a modification of δ_Q value has no effect on the output of the DCA, but it may have a significant impact on the content of the agent system in DYNAMO-MAS and, consequently, on the way the ontology is modified. In DYNAMO-MAS, the modification of δ_Q changes the relations and concepts considered as agents. In fact, the behaviors of *term* agents and *concept* agents depend on the selection of relations with the higher Q values, which are the one close to the maximum Q score. When an agent needs to select among several competing relations that could connect it with another agent, it will select the relation with the highest value for Q. When two relations have the same Q value, the agent selects the relation with the higher value for I (i.e., the larger number of occurrences). When δ_Q increases, more patterns and more relation occurrences are taken into account for this relation. So the concepts and semantic relations added to the ontology can be different when δ_Q is modified.

In practice, we did not find any case in which a term agent or a concept agent needed to use the I parameter to choose between two relations. As a perspective, we will study more precisely the influence of the modification of δ_Q on the quality of the evolved ontology.

In a similar way, the α_i parameters in the termAgRelevance and the conceptAgRelevance formula have been set after comparing the quality of different ontology evolution results. For the time being, the selected value for α_i produces the best results for ontology evolution in the three experimental domains. As a perspective, we want to introduce an Adaptive Value Tracker (AVT) component [23] to adjust the value of the α_i parameters in DYNAMO-MAS. An AVT is a software component that finds the optimal value of a dynamic variable in a given space thanks to successive feedbacks. In our case, feedbacks come from the ontologist. When the ontologist rejects or accepts a set of concepts, relations, and terms, the AVT will adjust α_i. The goal of the AVT is to make the relevance of refused terms and concepts lower than the proposal threshold while maintaining the relevance of accepted terms and concepts greater or equal to the proposal threshold. As a result, we will obtain more precise values of the α_i parameters for the evolved ontology.

Finally, in a general way, the evolution of ontologies from text faces some technical problems due to the limitations of current NLP tools. Indeed, when some terms have not been detected by the term extractor, the MAS fails to propose the corresponding concept.

5 Related Works

5.1 Evolving Ontologies from Text

When the DYNAMO project began (2007), we identified very few studies dealing with ontology evolution from text. Since then, several surveys about ontology evolution have been published [18,33]. Nevertheless, it seems that little attention has been paid to the automatic identification of knowledge (especially in text) in the perspective of ontology evolution.

Existing works dealing with ontology evolution refer mainly to the management of the evolution process after one or more changes (usually a manual one) or to the management and the comparison of versions [14,21]. These research works established an ontology of change types [21], the main stages of the evolution process [39], tools to detect change needs [12], to implement them [39], and to formally manage the consistency of the ontology after a change [13,39]. Some works refer to the propagation of changes to dependent artifacts of the modified ontology, i.e., to the objects, meta-data, ontologies, and applications that use this ontology [21,25,39]. Ontology evolution from text is addressed by two approaches: DYNAMO and EVOLVA.

EVOLVA [44] contributes in an original way to the evolution of ontologies from text within the NEON platform. As in our approach, the inputs of EVOLVA are the results of text analysis, i.e., a filtered list of candidate terms (single words) extracted from text. The main contribution of EVOLVA concerns the enrichment step that consists in finding a relation between new terms and concepts that already belong to the ontology. For this, EVOLVA uses ontologies already available on the Web, within which the system seeks a relation (direct or indirect) between a term to be added to the ontology and a node of the found ontology. To select the most relevant ontologies, one of the EVOLVA modules relies on patterns that look for common subgraphs between the ontology to be reused and the existing one. The terms for which relations have been found are added as new concepts in the ontology, and connected to existing concepts with the found relations. EVOLVA is effective for the English language (language for which some relevant terms extractors exist). Furthermore, the English language is used in the majority of the ontologies available on the Web. Concerning the French language, too few ontologies exist and semantic relations between new concepts and the ontology are hardly found. So EVOLVA is much less relevant to enrich ontologies in French.

In order to compare DYNAMO-MAS and EVOLVA, we have carried out an experiment with an English corpus [45] dealing with software bugs. EVOLVA has not been able to discover new concepts in this very specific domain and would have similar limitations in any very specific domain. The term extractor can identify new nouns, but no ontology can be reused to connect the new terms with the nodes in the ontology. Moreover, EVOLVA is not relevant for French corpora because there are too few French ontologies available on the Web. One main distinction between EVOLVA and DYNAMO-MAS is the way terms and concepts are managed. In EVOLVA, all extracted terms are suggested as concepts. In DYNAMO, terms and concepts are two distinct classes and all the extracted terms will not give birth to a concept. DYNAMO-MAS is able to work both with French and English corpus. It is also able to suggest new concepts and new terms when the ontology is very specific. To filter out the candidate terms, EVOLVA relies on relations extracted from Web ontologies, whereas DYNAMO-MAS uses linguistic and statistical criteria.

In the first DYNAMO prototype, agents implement a distributed clustering algorithm to organize terms into classes (concepts) and to structure these classes in a hierarchy [31,32]. DYNAMO-MAS is not an evolution of this first prototype. This prototype uses the results of the Syntex [7] syntactic parser that provides lists of terms and syntactic relations that have been found in the corpus. A hierarchical clustering algorithm, distributed in the agents, enables to move closer similar terms according to the principles of the distributional analysis, then to create and to position concepts. Two *term* agents are similar if they play a similar role (subject, verb, head or expansion) of a same set of terms. This MAS evolves until all agents are hierarchically related. The final state of the MAS corresponds to the ontology. The ontologist can validate it, refine it, or modify it. These actions are considered as perturbations by the MAS; as a feed-back, it reorganizes the agents and produces another ontology proposition. This interactive process is repeated until a satisfactory state of the ontology is obtained. Experimentations conducted with this prototype confirmed that statistical approaches are not effective when the volume of data in the corpus is small [27]. In the DYNAMO-MAS project, lexico-syntactic patterns are used for extracting lexical relations between terms and for establishing conceptual relations between concepts.

Using a MAS to represent an ontology is an originality of the DYNAMO project. Indeed, to our knowledge, no work exists (apart from the first DYNAMO prototype) that proposes a MAS to build and modify ontologies. In practice, ontologies are used to enable MAS to communicate [15,22,40]. Sometimes, to communicate, an agent uses its own ontology; it is able to modify it (for instance by replac-

ing a concept) according to its interactions with other agents [22]. Finally, some MAS have been used to align ontologies [43].

5.2 NLP Software Tools to Evolve Ontologies from Text

Although the evolution of an ontology deals with some specific and distinct problems from those related to ontology building, we relied on several works about building ontologies from text. Indeed, these approaches have allowed us to select some relevant techniques for texts analysis. Whether to build or to modify an ontology, we assumed that the documents of a given field contain the linguistic clues of relations and of concepts that need to be defined. More importantly, these studies provide criteria and methods to organize linguistic elements and to lead to the definition of classes or conceptual relations. We tried to identify among these criteria and techniques those that could be implemented in DYNAMO-MAS agents.

We do not propose here a new state of the art on existing tools in natural language processing (NLP) that can be used to modify ontologies from texts. Recent papers present such an overview [8], [29], or [11]. These tools can be useful to the "information extraction" process, although it is difficult to characterize the nature of the sought information. They often use lexical or semantic resources such as thesauri or lexical databases. Linguistic traces of concepts are generally assumed to be noun phrases, identified thanks to term extractors as well as on named entity extractors. Term classification techniques gather them into classes that define concepts from synonyms or similar terms. The selection of relevant terms among all extracted candidates is a heavy task. It can rely on several numerical criteria such as frequency, term productivity (its ability to be part of other phrases), its representativeness with regard to the corpus (with Tf.Idf), or by comparing its frequency in the corpus with its frequency in a reference domain independent corpus.

To define relations between concepts, two types of approaches exist. The first one is *statistical*. It is based on the information amount criterion in text. Relations between concepts are identified according to the co-occurrence[8] of terms or thanks to the shared context between words. However, statistical approaches are not able to provide an interpretation of these relations [19]. The second type of approach, more linguistic, involves the definition of *lexico-syntactic patterns* [20]. This approach has been implemented in [4,5,28].

5.3 Integrated Approaches to Evolve Ontologies from Text

Language analysis techniques enable to obtain linguistic clues on different kinds of knowledge that need to be filtered out and structured. Moreover, as a text collection does not cover all the knowledge of a domain, we obtain fragments of that do not always share concepts and that are complex to connect together. To overcome these two limitations, automated processing chains have been *a priori* defined, from text analysis to the production of a hierarchy of concepts. Statistical criteria enable to filter out and select the most relevant information. In addition, to connect the fragments between them, they are attached to a generic ontology. The most advanced tools are presented bellow:

- In *Text Onto Miner* (TOM) [16], the corpus is analyzed at different levels of granularity (document, paragraph, sentence), essentially according to statistical measures, to extract terms and lexical relations. Classification techniques are then used to propose a hierarchy of concepts.
- *OntoLearn* [42] uses a statistical approach to extract and to aggregate the terms of a corpus, and thus to build an ontology that specializes a generic ontology or WordNet. Extracted domain terms are gathered and linked to concepts from a generic ontology using similarity measures.
- similarly, *Text-To-Onto* [12] proposes to iteratively specialize a generic ontology or WordNet. Concepts and relations *sort_of* between concepts are discovered by automatic learning from examples of patterns and association rules. Finally, the ontology is pruned according to statistical measures and presented to the expert for evaluation.
- *DOGMA* [34] uses an approach similar to that of ASIUM, but more automated. Candidate terms are gathered according to their syntactic similarity, based primarily on the verbs encountered in text. The system builds a taxonomy of terms using a statistical approach that organizes classes of candidate terms.
- *CIAULA* [6] enables to build a hierarchy of verbs whose clusters are semantically close. Verbs are clustered thanks to a classification algorithm. The use of WordNet enables then to select, for each cluster, the best label among the names of WordNet synsets.

These platforms are only effective for large corpora and are not suitable for small-volume corpora such as those of DYNAMO. Moreover, the user of TextOntoMiner has to be strongly qualified in NLP techniques to organize the processes of text analysis and ontology building. Another difference is the attention given to the ontologist. In DYNAMO, we emphasize the notion of interactive ontology "joint-development", i.e., the ontologist can accept, reject, or modify the proposals made by the MAS; the MAS has then to integrate the ontologist decisions to provide new proposals that the ontologist has then to check again, and so on.

In previous approaches, the tools are a bit less interactive or not interactive at all; only the final ontology is proposed to

[8] Simultaneous presence of two or more terms in a window of words.

the ontologist. This postpones the validation. Furthermore, an existing generic ontology provides the missing knowledge. Finally, these works manage in a different way ontology building and ontology evolution, whereas we want to handle them uniformly.

6 Conclusion and Perspectives

The DYNAMO-MAS system contributes to assess the gains brought by the MAS paradigm for building and evolving ontologies from text. It differs from other tools by the following features:

1. It enables to make ontologies evolve: it supports an incremental evolution by taking into account new data (new text documents and interactions with the ontologist), and it avoids to restart the building process from scratch;
2. It is based on the MAS paradigm to make an ontology evolve: the originality comes from the fact that a part of the MAS is a bijective implementation of the ontology;
3. It makes an ontology evolve at runtime with the help of the ontologist. The ontologist can make corrections on the ontology that will be taken into account at runtime by the system;
4. The ontologies that are managed by DYNAMO-MAS are defined in OWL format, a standard that makes them easily reusable.

The experiments made with DYNAMO-MAS confirmed that linguistic clues are not sufficient to decide the content of an ontology and that the intervention of an ontologist is fundamental. Thus, some concepts proposed by the MAS are set under the *DomainThing* concept (ontology TOP concept). In fact, the knowledge of the involved agents is limited and does not allow them to be correctly set in the hierarchy of the ontology. In addition, to reduce the task of the ontologist, we are currently studying how to use other knowledge sources (including other ontologies, semantic dictionaries, etc.) that would enrich the knowledge of each agent and improve the DYNAMO-MAS results.

Two improvements are currently on-going: (i) the definition of behaviors enabling the agents to decrease the number of useless and wrong proposals; each agent is given the ability to detect its uselessness and (ii) the detection of more lexical relations using external dictionaries (such as WOLF and Wordnet) to improve each agent knowledge. The first results are very promising; they show that 56 % of the concepts and 67% of the terms proposed by the MAS were valid according to the ontologist.

In order to test DYNAMO-MAS in context, we will provide the system to the project partners so that they can use it for a long time period. We also want to experiment the quality of DYNAMO proposals when several ontologists modify various ontologies.

Acknowledgments We thank the ANR and all the DYNAMO project partners for their contribution to this work. We would particularly thank the industrial partners Sylvain Rougemaille (Unsolved Problems for Emerging Technologies - Upetec) and Mohamed Mbarki (Artal Technologies) for their contributions to the implementation of the DYNAMO tool.

References

1. Aubin S, Hamon T (2006) Improving term extraction with terminological resources. In: Salakosk T, Ginter F, Pyysalo S, Pahikkala T (eds) Advances in natural language processing 5th international conference on NLP. Springer, Turku, pp 380–387
2. Aussenac-Gilles N, Condamines A, Sèdes F (eds) (2006) Ressources termino-ontologiques. Revue I3 (Information–interaction–intelligence). Cépaduès-edn
3. Aussenac-Gilles N, Despres S, Szulman S (2003) The TERMINAE method and platform for ontology engineering from texts. In: Buitelaar P, Cimiano P (eds) Bridging the gap between text and knowledge—selected contributions to ontology learning and population from text. IOS Press, Amsterdam, pp 199–223
4. Aussenac-Gilles N, Séguéla P (2000) Les relations sémantiques : du linguistique au formel. Cahiers de Grammaire. "Sémantique et Corpus". Textes réunis par A. Condamines 25:175–198
5. Barrière C, Akakpo A (2006) Terminoweb: a software environment for term study in rich contexts. In: Proceedings of international conference on terminology, standardization and technology transfer. Encyclopedia of China Publishing House, Beijing, pp 103–113
6. Basili R, Velardi P, Pazienza MT (1996) Integrating general-purpose and corpus-based verb classification. Comput Linguistics 22(4): 559–568
7. Bourigault D (2007) Un analyseur syntaxique opérationnel : Syntex. PhD Thesis, Univ. de Toulouse Le Mirail, France. Habilitation à diriger des recherches
8. Buitelaar P, Cimiano P, Magnini B (2005) Ontology learning from text: methods evaluation and applications. Frontiers in artificial intelligence and applications series, IOS Press, Amsterdam
9. Camps V(1998) Vers une théorie de l'auto-organisation dans les systèmes multi-agents basée sur la coopération : application à la recherche d'information dans un système d'information répartie. Thèse de doctorat, Univ. Paul Sabatier, Toulouse, France
10. Cimiano P (2006) Ontology learning and population from text: algorithms evaluation and applications. Springer, Berlin
11. Cimiano P, Buitelaar P, Völker J (2010) Ontology construction. In: Indurkhya N, Damerau FJ (eds) Handbook of natural language processing, 2nd edn. Taylor and Francis Group, Boca Raton
12. Cimiano P, Völker j (2005) Text2onto—a framework for ontology learning and data-driven change discovery. In: Montoyo A, Munoz R, Metais e (eds) Proceedings of the 10th international conference on applications of natural language to information systems (NLDB). Springer, Heidelberg, pp 227–298
13. Flouris G(2006) On belief change and ontology evolution. Ph.D. thesis, University of Crete, Department of Computer Science, Heraklion, Greece
14. Flouris G, Plexousakis D, Antoniou G (2006) A classification of ontology change. In: Poster session of the 3rd Italian semantic web workshop, semantic web applications and perspectives (SWAP-06), CEUR Workshop proceedings, vol 201. CEUR-WS.org
15. Gandon F (2002) Distributed artificial intelligence and knowledge management: ontologies and multi-agent systems for a corporate

semantic web. Thèse de doctorat, Univ. de Nice - Sophia Antipolis, Nice, France
16. Gawrysiak P, Protaziuk G, Rybinski H, Delteil A(2008) Text onto miner: a semi automated ontology building system. In: Proceedings of the 17th international conference on foundations of intelligent systems, ISMIS'08, Springer, Berlin, pp 563–573
17. Gleizes MP, Camps V, Georgé JP, Capera D(2007) Engineering systems which generate emergent functionalities. In: Engineering environment-mediated multi-agent systems—satellite conference, The European Conference on Complex Systems. Katholieke Universiteit Leuven, Dresden
18. Haase P, Sure Y, Vrande D (2004) D3.1.1 ontology management and evolution—survey, methods and prototypes. Eu-ist ip ist-2003-506826 sekt, project deliverable, Institute AIFB, University of Karlsruhe, Karlsruhe
19. Harris ZS (1968) Mathematical structures of language. Wiley, New York
20. Hearst MA (1992) Automatic acquisition of hyponyms from large text corpora. In: 14th internaitonal conference on computational linguistics (COLING), pp 539–545
21. Klein M (2004) Change management for distributed ontologies. Ph.D. thesis, Dutch Graduate School for information and knowledge systems, Germany
22. Leen-Kiat S (2002) Multiagent distributed ontology learning. In: Cranefield S, Finin, S. Willmott (eds.) Proceedings of the Workshop on Ontologies in Agent Systems, 1st Int. Joint Conference on AAMAS. Vol 66Bologna, Italy, pp 75–79
23. Lemouzy S (2011) Systèmes interactifs auto-adaptatifs par systèmes multi-agents auto-organisateurs : application a la personnalisation de l'accès à l'information. UniversitT Paul Sabatier, Toulouse, France, ThFse de doctorat
24. Levenshtein VI (1966) Binary codes capable of correcting deletions insertions and reversals. Soviet Phys Doklady 10:707
25. Luong PH (2007) Gestion de l'évolution d'un web sémantique d'entreprise. Ph.D. thesis, Ecole doctorale STIC, école des mines de Paris, Paris
26. Maedche A (2002) Ontology learning for the Semantic Web. Vol 665, Kluwer Academic Publisher, Dordrecht
27. Malaise V (2005) Méthodologie linguistique et terminologique pour la structuration d'ontologies différentiellespartir de corpus textuels. Ph.D. thesis, Paris 7 Denis Diderot University
28. Morin E (1999) Using lexico-syntactic patterns to extract semantic relations between terms from technical corpus. In: 5th international congress on terminology and knowledge engineering (TKE). TermNet, Innsbruck, pp 268–278
29. Nédellec C, Nazarenko A, Bossy R (2009) Information extraction. In: Staab S, Studer R (eds) Ontology handbook. Springer, Berlin, pp 268–278.
30. Noel V, Arcangeli JP, Gleizes MP (2010) Component-based agent architectures to build dedicated agent frameworks (regular paper). In: Trappl R (ed) international symposium "From Agent Theory to Agent Implementation" (AT2AI). Austrian Society for Cybernetic Studies, Vienna
31. Ottens K (2007) Un système multi-agent adaptatif pour la construction d'ontologies-partir de textes. Thèse de doctorat, Univ. Paul Sabatier, Toulouse, France
32. Ottens K, Hernandez N, Gleizes MP, Aussenac-Gilles N (2008) A multi-agent system for dynamic ontologies. J Logic Comput 19:1–28
33. Palma A, Haase P, Wang Y, d'Aquin M (2007) D1.3.1: Propagation models and strategies. Technical report, NeOn IST Project Deliverable
34. Reinberger ML, Spyns P (2004) Discovering knowledge in texts for the learning of dogma-inspired ontologies. In: Proceedings of the workshop Ontology learning and population, ECAI04, Valencia, Spain, pp 19–24
35. Reymonet A, Thomas J, Aussenac-Gilles N (2007) Modelling ontological and terminological resources in OWL DL. In: Buitelaar P, Choi KS, Gangemi A, Huang CR (eds) OntoLex07—from text to knowledge: the Lexicon/Ontology interface, Workshop at ISWC07 6th International Semantic Web Conference, Busan
36. Sellami Z, Aussenac-Gilles N, Gleizes MP, Camps V (2012) DYNAMO : un outil de construction et d'évolution d'ontologies partir de textes. Technique et Science Informatiques 31/1, pp 97–124
37. Sellami Z, Camps V (2012) Evaluation of a multi-agent system for the evolving of domain ontologies from text (regular paper). In: International conference on practical applications of agents and multiagent systems (PAAMS), Salamanca, 28/03/12-30/03/12, 4240, vol 155. Springer, Berlin, pp 169–179 http://www.springerlink.com
38. Sellami Z, Gleizes MP, Aussenac-Gilles N, Rougemaille S (2009) Dynamic ontology co-construction based on adaptive multi-agent technology. In: International conference on knowledge engineering and ontology development (KEOD), INSTICC
39. Stojanovic L (2004) Methods and tools for ontology evolution. Ph.D. thesis, Karlsruhe University, Germany
40. Tamma V, Bench-Capon T (2002) An ontology model to facilitate knowledge-sharing in multi-agent systems. Knowl Eng Rev 17:41–60. doi:10.1017/S0269888902000371
41. Tissaoui A, Aussenac-Gilles N, Laublet P, Hernandez N (2010) Gestion des évolutions d'une ressource termino-ontologique et des annotations sémantiques. In: Aussenac-Gilles N, Hernandez N, Laublet P (eds) Atelier "Evolution d'ontologies" des 21èmes Journées francophones d'Ingénierie des Connaissances (IC2010). Nîmes, France
42. Velardi P, Navigli R, Cucchiarelli A, Neri F (2005) Evaluation of ontolearn, a methodology for automatic learning of domain ontologies. In: Buitelaar P, Cimiano P, Magnini B (eds) Ontology Learning from text: methods. Evaluation and applications. IOS Press, Amsterdam
43. Wang J, Les Gasser L (2002) Mutual online ontology alignment. In: Cranefield S, Finin T, Willmott S (eds) Proceedings of the Workshop on Ontologies in agent systems, 1st international joint conference on AAMAS. Vol 66, Bologna, Italy, pp 103–113
44. Zablith F, Sabou M, d'Aquin M, Motta E (2009) Ontology evolution with Evolva. In: L.e.a. Aroyo (ed) Proceedings of the 6th European semantic web conference (ESWC). Vol LNCS 5554, Springer, Berlin, pp 908–912
45. Zablith F, Sellami Z, D'Aquin M, Aussenac-Gilles N, Hernandez N (2010) Vers la combinaison de deux techniques d'évolution d'ontologies à partir de ressources générales et de ressources linguistiques. In: Atelier "Evolution d'ontologies" des 21e Journées francophones d'Ingénierie des Connaissances (IC2010), Nîmes