Dynamic Ontology Co-Evolution from Texts: Principles and Case Study

Kévin Ottens[1], Nathalie Aussenac-Gilles[1], Marie-Pierre Gleizes[1], Valérie Camps[1]

[1] Institut de Recherche en Informatique de Toulouse
Université Paul Sabatier
118, Route de Narbonne 31062 Toulouse cedex 9 – France
{ottens/aussenac/gleizes/camps}@irit.fr

Abstract. As claimed in the Semantic Web project, a huge amount of physically distributed interacting software agents could find the semantic of available resources and answer more relevantly to users' requests if the content of these resources would be represented with formal semantic concepts defined in ontologies. Because Web information sources are highly dynamic and conceptually heterogeneous, one of the most challenging problems in the Semantic Web research is the proper and frequent ontology updating in keeping with knowledge changes. To tackle this problem, we have developed a self-organizing multi-agent system -Dynamo- able to create an ontology draft from automatic text processing. Because it is well-known that only a part of a domain description is explicitly described in texts, Dynamo enables an ontology co-construction with a domain expert in a fully interactive way. In this paper, we present the principles of this approach and related experiments.

Keywords: Collaborative ontology construction from text, adaptive multi-agent system, ontology dynamics, ontology maintenance.

1 Introduction

The challenge of an efficient information retrieval on the Web requires to define relevant resources for document tagging and indexing. Two apparently competitive trends emerged: whereas the Semantic Web [1] suggests the use of normalized and formal concepts in ontologies defined by domain specialists, the Web 2.0 tools make it possible to collaboratively organize and share hierarchies of possible tags. These two trends offer complementary features. Their combination could benefit both of the precision and formalism of ontologies, and of the fast reactivity and the powerful collaborative effort that lead to build Web2.0 lists of tags. Recent investigations propose to rely on the strengths of these two trends, mainly to get updated resources that match the evolution of knowledge sources on the Web.

Indeed, ontologies are rigid structures that are difficult to update. When used in Semantic Web applications, they are immerged in a highly dynamic environment, where new and conceptually heterogeneous information sources appear every day. Domain specific and technical knowledge is more prone to change than expected. An

attempt to evaluate this dynamics [2] has shown that ontology maintenance is now one of the key issues for their use in Web applications: "*The only feasible approach for dealing with dynamic domains is speeding up ontology maintenance. It is obvious that monthly or weekly updates of the ontologies in our simulation experiments will drastically reduce the amount of missing elements*". So far, one of the major challenges for the Semantic Web research is the proper and frequent ontology updating in keeping with knowledge changes.

These changes could come from the integration of tags list built in Web 2.0 collaborative applications, from the integration of new Web sites and databases, or from manual modifications proposed by experts. To tackle this problem, we propose to combine the recent advances in ontology learning from texts with the help of Natural Language Processing tools and the flexibility of adaptive agent programming. We have developed a self-organizing multi-agent system - Dynamo[1] - able to create and maintain an ontology draft from automatic text processing. As long as only a part of domain knowledge is explicitly described in texts, Dynamo expects domain experts to add missing knowledge to this draft and to interact with the system until they get a satisfying ontology. This system assumes that ontology engineering is a continuous cycle where texts or humans may suggest some modifications. In this paper, we present the principles of ontology co-construction with Dynamo and some validation experiments of the approach.

First, we briefly describe works related to ontology construction and maintenance from texts. Section 2 expounds the basic principles of the distributed Dynamo algorithm that creates a draft ontology from text. This algorithm is implemented with a multi-agent system where the agents are the concepts of the ontology running to discover their right place inside the organization. Section 3 illustrates with an example the process of ontology creation from text. This is a co-construction process where the ontologist and Dynamo interact in real-time according to their respective knowledge. Properties of this software are analysed with regard to this experiment in section 4 before concluding in section 5.

1.1 Ontology Engineering from Texts: Short Overview

Ontology engineering from texts has reached enough maturity to be considered as an efficient way to build ontologies, with the extra advantage that various lexical forms can be obtained for each concept. Recent books like [3] and [4] provide a good overview of existing methods and tools. They illustrate the diversity of techniques that can be applied to get various kinds of specific linguistic evidences of domain knowledge. These syntheses confirm the necessity to combine linguistic and statistical approaches to text mining with different perspectives, like term extraction, semantic class identification, relation extraction, … Whatever the quality of the tools and the relevance of their combination may be, only a part of an ontology can be learned from text: results of the learning process generally are called *draft* or *kick-off ontologies* [5]. They need to be formalized and their ontological properties have to be checked.

[1] DYNAMO is an acronym for « DYNAMic Ontologies »

Nevertheless, only a few methods, like Text2Onto [6], have paid attention to ontology maintenance by using Natural Language Processing. The Text2Onto framework helps to semi-automatically learn and update ontologies from domain specific texts by applying machine learning techniques [6]. [7] uses a neural network system for term extraction and latent segment analysis for term clustering and incremental concept identification. In both cases, the authors underline the need for these tools to provide facilities for a manual engineering of the learned network. Only human intervention can guarantee that the ontology fulfils the application requirements. Maedche and Staab call it *balanced cooperative modelling* [8].

1.2 Statements underlying Dynamo

Our contribution follows this paradigm. Our system, Dynamo, can be used to build ontologies or to maintain them. The current system is able to maintain only Dynamo designed ontologies. But the target is to be able to dynamically update existing models with the knowledge learned from texts. We focus mainly on term extraction as a means to identify domain concepts, and on term clustering based on their syntactical structure to learn hierarchical relations. In our approach, term extraction is carried out by an independent tool, the Syntex system [9], that runs syntactical and distributional analyses. Dynamo defines an adaptive multi-agent system (MAS) from each terminological network provided by Syntex and the available agents that form the ontology to be maintained. These agents organise themselves so that they form a hierarchy of concepts. We consider this hierarchy as the resulting draft ontology. Because it combines a conceptual network and related terms, we call it a *termino-ontological resource*.

The organization process relies on a clustering algorithm, detailed in [10], the originality of which is to be distributed over all the agents. Although its design is inspired by classical agglomerative hierarchical clustering [11], this algorithm tends to break up clusters locally identified by each agent. Inputs are the candidate terms provided by Syntex, and it exploits syntactical relations between terms to define clusters. The major gain brought by this new implementation is that feed-back can be manually provided before the clustering is completed, which makes it possible to understand and modify the obtained clusters. This MAS enables the dynamic construction of a class hierarchy from an entry data flow. Each node of the hierarchy is a concept-agent created when a new term is taken into account. An agent's behaviour enables to merge it with a sibling agent or to raise one of its child agents, according to a similarity measure locally computed. The resulting classification is the hierarchy of the multi-agent system itself. As we will see in section 2, this agent's behaviour is not sufficient to create an ontology or even a taxonomy for two reasons: there is no rule to simplify the hierarchy and this is no multi-criteria algorithm.

2 Ontology as a Self-Organizing Multi-Agent System

Dynamo is a tool, based on an Adaptive Multi-Agent System (Amas), enabling the construction and the maintenance of an ontology starting from a textual corpus. Multi-

Agent Systems provide solutions to problems involving several autonomous entities (called "agents") which can be geographically and logically distributed, which are plunged into a dynamic environment, which have a partial perception of this environment, and which have limited cognitive capacities. More precisely, the aim of Dynamo is to build a draft domain specific termino-ontological resource (the multi-agent system or MAS). This draft is a hierarchy of concepts which results from the MAS organisation where each concept is represented by an agent. Dynamo is a semi-automatic tool because the ontologist[2] has to validate, refine or modify the hierarchical organization of concepts until it reaches a satisfying state. The Dynamo system consists of three parts:

- a network of terms, obtained with the Syntex term extractor [9] from a textual corpus. Syntex runs a dependency structure analysis to extract all possible candidate-terms from a corpus (in French or in English); it relies on head-expansion relations between compound terms to organize them into a network[3]; and it runs a distributional analysis in order to suggest classes of terms that share similar syntactic contexts. Each term is given in its lemmatized form, with a list of all its occurring sentences, its head term and expansion terms, related terms that share similar use contexts and statistics (frequency, productiveness …). Syntex has been used many times for ontology building;
- a multi-agent system which carries out a hierarchical clustering over the term network and produces a taxonomy of concepts. Agents composing the system cooperate to position themselves in a hierarchy and the multi-agent system constitutes the resulting taxonomy. When building a net ontology, this network is empty at first, but when maintaining an ontology built with Dynamo, it contains the current network of concept-agents.
- an interface enabling the ontologist to visualize and control the clustering process, and to modify the resulting hierarchy.

Our approach to create an ontology as the result of self-organising process in a MAS is, to our knowledge, completely original. "*Self-organisation is the mechanism or the process enabling a system to change its organisation without explicit external command during its execution time*" [12]. This choice comes from the qualities offered by this kind of multi-agent systems: they make easier the interactive design of a system (in our case, a conceptual network), they enable its incremental building by progressively taking into account new data (coming from text analysis and user interaction), and, last but not least, they can be easily distributed across a computer network. With this approach, ontology is seen as a stable network composed of conceptual entities, here represented by "concept agents", linked with labelled relations. Another advantage over a centralized clustering algorithm is that results of intermediate steps can be checked and corrected.

The distributed clustering algorithm implemented with an Amas (whose principle and evaluation are explained in [13]) tends to introduce new layers in the taxonomy. It

[2] We call an *ontologist* a knowledge engineer or an analyst, in charge of building an ontology from knowledge sources.

[3] For instance, the term *knowledge acquisition* has the term *acquisition* as head and *knowledge* as expansion, and it is the expansion of the term *knowledge acquisition system*.

is designed to be both the system producing the resulting structure and the structure itself. It means that each agent represents a concept and its autonomous and cooperative behaviour is to find its right place in the organization, namely in the taxonomy. Each agent possesses communication capacities and behaviours to structure and modify the taxonomy according to different rules. The system output is the organization obtained from the interaction between agents, while taking into account feedback coming from the ontologist when he modifies the taxonomy according to the application requirements or his expertise.

Furthermore, the agents' behaviour rules enable several organisational modifications locally by taking into account their parents/child relations. These local modifications are listed in the three following points:

1. The "head coverage" rule tends to push involved agents toward the leaves of the taxonomy. To do that, each agent determines if its parent is adequate. This is possible because each concept agent is described by a set of terms that belong to the head-expansion term network. If T_X is the set of terms describing a concept agent X and head(T_X) the set of all the terms that are head of at least one item of T_X, the parent adequacy function a(P,C) between a parent P and a child C can be defined by the following formula : $a(P,C) = |T_P \cap head(T_C)| / |T_P \cup head(T_C)|$. Then, the best parent for C is the agent P that maximizes a(P, C).
   Rule1: when agent C is unsatisfied with its parent P, it evaluates a(B_i, C) with all its siblings (noted B_i); the one maximizing a(B_i, C) is chosen as the new parent.

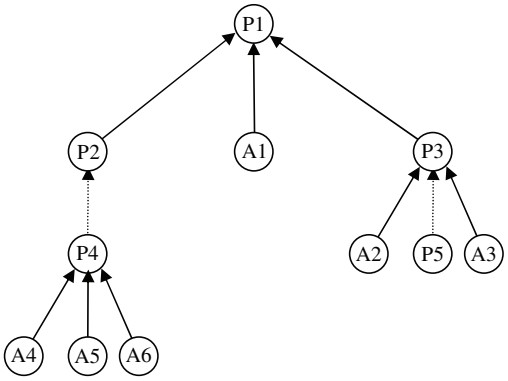

Figure 1. Simplification branch and uselessness rules

2. The "simplification branch and uselessness" rules force the agent to go up the hierarchy, as shown in the figure 1.
   Rule2: When an agent has several children but no sibling (like P_4), then it proposes to its children (A_4, A_5 and A_6) to have its own parent (P_2) as new parent.
   Rule3: When an agent has no children and is represented by no term (like P_5), it has to leave the system.
3. The "similitude tolerance" rules enable to obtain n-ary trees forcing the agent to go up the hierarchy and to simplify the structure by aggregation. More precisely, with

the distributed clustering algorithm and the previously presented rules, the result of the Amas is necessarily a binary tree (unless for the last level of the hierarchy if the rule 1 has been applied). The hierarchy resulting from this basic algorithm is always a binary tree because this algorithm separates items when similarity is different from 1.0 in order to form clusters. But our objective is to obtain a dynamic taxonomy, which is rarely a binary tree. To obtain n-ary nodes rather than binary nodes, each concept agent A introduces a tolerance ε in its vote (which follows the Condorcet vote strategy), and only keeps its vote for its siblings F_k such as $1 - sim(A, F_k) > \varepsilon$. This tolerance is locally managed by each concept agent; it takes into account the tolerance value of its parent and its own tolerance in order to influence the connection factor. The ontologist can give to the system an interval for the global connection factor and each concept agent has then to adjust its local tolerances to try to conform to this interval while taking into account dissimilarities with its neighbourhood. Two rules have been defined to take into account these tolerance variations.

Rule4: When an agent P_0 has its children which do not enforce any more the property about tolerance ε_{P0}, then P_0 proposes to its children to have its parent P as new parent.

Rule5: When an agent P_0 has a number of children too high (resp. too low), it decreases (resp. increases) its tolerance ε_{P0}.

Each concept agent has to deal with multiple criteria during the taxonomy building and has to determine its priorities at a given time. More precisely, each concept agent computes three non cooperation degrees and chooses its current priority according to the highest one. This priority is used during message passing and each message possesses a priority p_k corresponding to the non cooperation degree of the agent when it sends it. For an agent A having a parent P, a set of siblings B_i and which received a set of messages M_k having the priority p_k, the three computed non cooperation degrees are:

- $\mu_H(A) = 1 - a(P, A)$, is the "head coverage" non cooperation degree, determined by the head coverage of the parent,
- $\mu_B(A) = \max(1 - \text{similarity}(A, B_i))$, is the "siblings" non cooperation degree, determined by the worst sibling of A regarding similarities,
- $\mu_M(A) = \max(p_k)$, is the "message" non cooperation degree, determined by the most urgent message received.

The non cooperation degree of agent A is $\mu(A) = \max(\mu_H(A), \mu_B(A), \mu_M(A))$. Then, we have three cases determining which kind of action A will choose:

- if $\mu(A) = \mu_H(A)$ then A will use the head coverage rule (rule1) previously detailed;
- if $\mu(A) = \mu_B(A)$ then A will run the distributed clustering algorithm [13];
- if $\mu(A) = \mu_M(A)$ then A will process M_k immediately in order to help its sender.

Those three cases summarize the current activities of the agents: they have to find the best parent for them when $\mu(A) = \mu_H(A)$; they have to improve the agent network structure through clustering when $\mu(A) = \mu_B(A)$; or they have to process other agent messages when $\mu(A) = \mu_M(A)$ in order to help them to fulfil their own goals.

In this approach, we consider an ontology as a dynamic equilibrium between its concept agents. The ontology modification is a perturbation of the previous

equilibrium caused by the appearance or disappearance of concept agents or relationships. In this way, a dynamic ontology is a self-organizing process occurring when new texts are included into the corpus, or when the ontologist interacts with it.

3 Co-Evolution Experiment

Dynamo has been experimented to create an ontology draft from a corpus of abstract scientific English papers published in the journal «Astronomy and Astrophysics» edited by Springer (http://www.springerlink.com/content/300419/). An ontology had been created from this corpus by experts and a sub-part of it (approximately one hundred nodes) will be the reference for evaluating our work. Because both the reference ontology and the learned one are designed from the same documents, we assume that their terminologies are overlapping. The experiments aim at showing the dynamic evolution relevance of the MAS. The set of parameters used during this experiment is composed of three elements:

1. The formula given in [14] to compare the similarity between two terms t1 and t2. This formula uses a, b, c and d which are respectively the number of contexts in which t1 and t2 are both present, only t1 (respectively only t2) is present and contexts where neither t1 nor t2 is present. The parameter α giving the balance between these contexts is fixed to 0,75.

$$sim(t1,t2) = \alpha\ /2\ *\ (\ a/(a+b)\ +\ a/(a+c)\ )\ +\ (1-\alpha)/2\ *\ (\ d/(d+c)\ +\ d/(d+b)\ )$$

2. The branching factor given by an interval [minValue, maxValue] which defines the number of children that a given concept could have in the graph. For example, if the maxValue is 2, we obtain a binary tree. According to his knowledge about astronomy, the ontologist has defined the branching factor interval as [2, 7].
3. We must also compare the obtained ontology from the experiment with the reference ontology. We used the measure of taxonomy overlapping given in [8] which defines a value between 0 and 1. When two taxonomies are identical, the corresponding measure is 1. This measure takes into account hierarchical relations and assumes that concepts with the same label are identical. So a low score of this measure means that the structure of both ontologies is very different.

3.1 Automatic Draft Ontology Creation from the Corpus

The system initializes the graph ontology by creating its root with the agent called TOP. Each term extracted from the corpus is then embedded into a corresponding agent concept linked with TOP. From this initial network of agent-concepts, the behaviour of each agent is launched according to the rules defined in section two. Each agent behaves in parallel by processing the local information coming from its neighbours. The self-organizing process of agents leads to a global equilibrium which corresponds to the initial draft ontology.

The result shown in figure 1 is then presented to the ontologist (for visibility reasons, we have suppressed some leaf concepts). Five main subsets found by

Dynamo have been highlighted with grey-blue dotted lines. They are identified as a root group, a main branch and three sub-branches.

The modifications carried out later on by the ontologist (see sections 3.2 to 3.5) will be considered as local perturbations by the concerned concept-agents. These agents will use again their behaviour in order to find a more cooperative location inside the organization (the ontology).

Figure 1. Resulting taxonomy from an autonomous resolution on the astronomy corpus

3.2 First Ontologist's Intervention

The similarity measure between the resulting taxonomy and the reference one gives a value of 0.78. This quite high value is mainly due to the good location of leaf concepts. Nevertheless, the global structure is unbalanced for the ontologist because the root graph contains three groups (sub-branch 1, 2 and 3) without clear semantics. Consequently the ontologist modifies the organization in bringing back these three groups (corresponding to ConceptAgent93, ConceptAgent94 and ConceptAgent97) to the root (he links them to the TOP concept hidden inside the "groupe racine").

These perturbations lead to a reaction of the considered concepts that re-evaluate their cooperation degree with their neighbours. Dynamo does about thirty link modifications, which leads to the new ontology draft shown in figure 2.

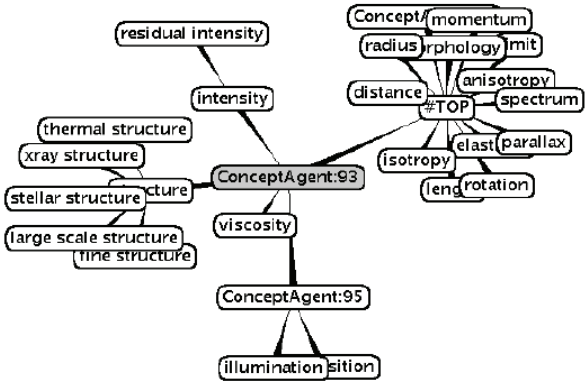

Figure 2. Resulting taxonomy after the first reorganization

In reaction to the changes made by the ontologist, Dynamo carries out the more important changes around ConceptAgent:93. These relevant changes enable to identify thematically ConceptAgent:93 as representing mechanical and thermo-dynamical properties. Moreover, the geometrical properties are directly linked to the TOP concept. This new structure obtains a value of 0.80 when compared to the reference ontology.

3.3 Second Ontologist's Intervention

Now the ontologist wants to improve the separation of the different emergent properties during his second intervention. His work consists in connecting thirteen concepts related with mechanical and thermo-dynamical properties to ConceptAgent:93. The concerned concepts are isotropy, morphology, momentum, anisotropy, spectrum, elasticity, mass, sensitivity, emission, density, entropy, diagram and temperature.

According to the behaviour rules, Dynamo moves then thirty concepts in the organization. The resulting structure contains now a complex branch describing the mechanical and thermo-dynamical properties. These actions by the ontologist moved the ontology away from the reference one and the similarity measure falls down to 0.76; this value will remain constant until the end.

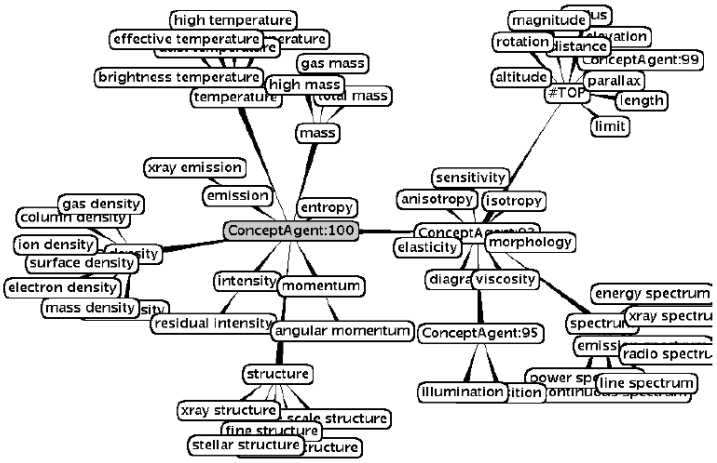

Figure 3. Organization after the second interaction between the ontologist and Dynamo

3.4 Third Ontologist's Intervention

The ontologist focuses now his work on the geometrical and optical properties found under ConceptAgent:99. He moves all the optical properties under the TOP concept, whereas some geometrical properties are linked to ConceptAgent:99.

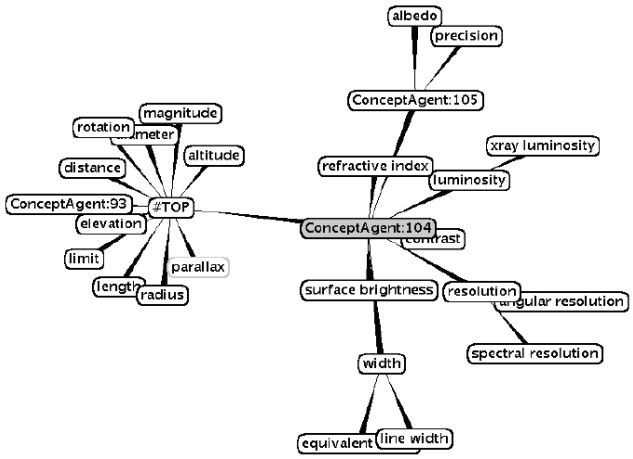

Figure 3. Third adjustment of the ontology

Dynamo reacts by moving all the geometrical properties under TOP and creates a new branch under TOP containing the optical properties and the "width" concept. This corresponds to a dozen of reorganizations processed by Dynamo.

The ontologist agrees with the destruction of the geometrical branch because he considers that the concepts previously aggregated had a high disparity. Thus, he keeps this change where geometrical properties were brought closer to thermo-dynamical and optical properties. Now, three coherent sets of concepts are linked to TOP: (*i*) the one directly connected under TOP; (*ii*) a group under ConceptAgent:93; (*iii*) a group under ConceptAgent:104.

3.5 Fourth Ontologist's Intervention

The remaining problem is the presence of the "width" concept under the optical branch (ConceptAgent:104). Thus, the ontologist moves it directly under TOP.

After the last self-organizing process of Dynamo (around ten link changes), we observe on the left a group about mechanical properties, in the middle a group about geometrical properties, and the optical properties on the right.

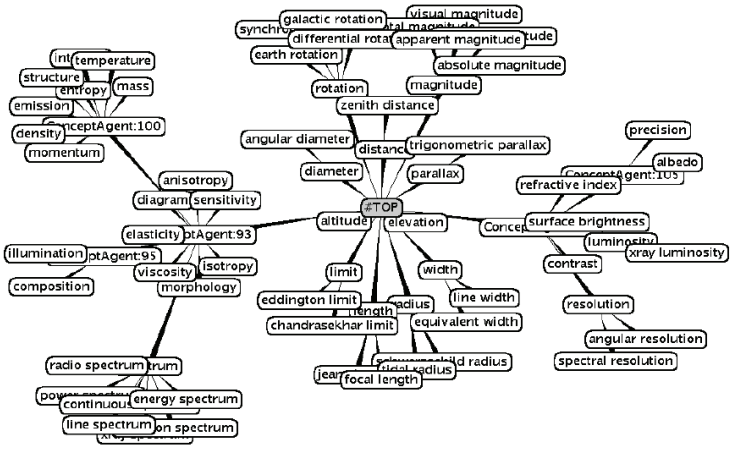

Figure 4. Final co-evolving draft ontology

4 Evaluation Analysis

In this section, the evaluation is twofold: a quantitative evaluation which relies on performance results and a qualitative one which is made by the ontologist. We then give the main perspectives of this work.

4.1 Quantitative Evaluation

As explained with details in [13], the complexity measures of the Dynamo system give interesting results. We carried out this work roughly in two phases. Firstly, we determined that the theoretical complexity of the core distributed clustering algorithm is $\mathcal{O}(n^3)$ like the simplest centralized clustering algorithm. But, in practice our experiments showed a $\mathcal{O}(n^2 \text{Log } n)$ complexity on average with a very good stability of the system due to its low behaviour variation across our data sets.

Secondly, we computed the average complexity of the whole system (that is the core clustering algorithm and the extra rules presented in this paper). Then the experimental complexity raised to $\mathcal{O}(n^3)$, once again with a very low variation and then a good stability of the system. Of course, this raise in complexity is explained by the more refined result obtained as output of the system. The system does more computations in this case, but the complexity of the whole still stays acceptable.

4.2 Qualitative Evaluation

The time spent by the ontologist to co-construct the final draft ontology is around three hours, including the great part needed for the difficult handling of the visualization interface. He estimates that using a traditional tool would have required up to four times the time he spent with Dynamo. The number of modifications that he brought to the taxonomy is quite manageable: 3 during the first step, 13 during the second step, about 15 during the third one and only one at the last step.

The resulting taxonomy is even more refined than the reference one, which explains the sub-optimal similarity measure. Moreover, Dynamo reduces the cognitive load of the ontologist because he may focus on the hardest difficulties. Once he has modified these concept-agents, the system propagates the consequences of these changes on related agents. For example, in the experiment, Dynamo has relevantly modified the edges in the graph five times more than the ontologist did.

A possible new experiment could be to follow the method proposed in [15] to improve the evaluation of the learned ontology with regard to the reference ontology.

4.3 Improvement and Future Work

As in any software prototype, several features of Dynamo could be improved, mainly if we want to update existing ontologies in Dynamo. We will focus here only on the two most important ones: user-friendliness and link labelling.

The first limitation of Dynamo comes from the lack of user-friendliness of the end-user interaction. Even with a restricted ontology size, the ontologist has great difficulties in following the dozens of link modifications done autonomously in only few seconds at each step. He must spend a lot of time localizing in the graph display the concepts that he has previously worked on. Indeed, only a small perturbation made by the end-user can potentially have important repercussions on the structure. An efficient ontology maintenance system would require a deep cooperation with ergonomists to define an easy-to-use graphical interface.

The second limitation comes from the created links which are labelled only with "is-a". Consequently, the current Dynamo prototype produces taxonomies and not full ontologies. The main reason for this is that we focussed in priority on head-expansion relations between terms, and their most frequent meaning is a hyperonymy relation between a term and its compound terms. But the pre-processor Syntex is able to provide some linguistic clues to define other semantic relations and their labels. Moreover, results from other natural language processing tools could be given as input to the agents. In the near future, we envisage two complementary techniques to select proper labels for relations between concepts (or links among agents):

– The instantiation of a predefined set of patterns (for example [X 'take' Determiner Adjective Y]) defined by the ontologist in a given domain. These patterns could be used by pattern-agents inside Dynamo and their work would be to scan the taxonomy in order to fill in these empty generic patterns with relevant candidates.
– The automatic creation of these patterns based on the correspondence between the relationships between terms (given by Syntex) and links in the ontology which are not labelled yet. This allocation problem will be solved by using the Amas technology.

5 Conclusion and Perspectives

Ontology maintenance is a challenge that we propose to consider in continuity with ontology construction. In this paper, we presented a new approach based on multi-agent technology in order to reduce the ontologist's amount of work, by creating autonomously concepts and their relationships from text extracted candidate terms. Dynamo is an innovative tool for dynamic ontologies from two points of view:

- First, at any time, new sets of documents can be added to the input corpus, new knowledge can be manually provided by the ontologist, leading to concept and relation additions or deletions. The system adapts the previous network according to this new information; thus, the ontology can be effectively dynamically updated.
- Second, the system and the ontologist modify the same network in a cooperative way: this process relies heavily on the strong coupling between the action of one of them and the reaction of the other.

The semi-automatic ontology construction from texts eases greatly the ontologist's work. Nevertheless, based on our experience, there are a lot of implicit relationships which cannot be discovered in analyzing a corpus of texts. Thus, even in increasing greatly the computer work, the final decision remains to the human [16]. For this reason, we agree with the design requirements for ontology evolution defined by [17]:

1. *It has to (i) enable resolving the given ontology changes and (ii) ensure the consistency of the underlying ontology and all dependent artifacts;*
2. *It should be supervised allowing the user to manage changes more easily;*
3. *It should offer advice to user for continual ontology refinement.*

We think that a collective agent process -like Dynamo- is a good way to be consistent with these requirements.

References

1. Benjamins, V.R., Contreras, J., Corcho, O., Gómez-Pérez A.: Six Challenges for the Semantic Web. *Proceedings of the Semantic Web workshop held at KR-2002*, Toulouse, France (2002)
2. Hepp, M.: Ontology Maintenance: Quantifying the Conceptual Dynamics in Domain Ontologies. *DERI Technical Report SEBIS-TR2007-01* (2007)
3. Buitelaar, P., Cimiano, P., Magnini, B., editor(s), *Ontology Learning from Text: Methods, Evaluation and Applications*, Frontiers in Artificial Intelligence, (123) IOS Press,2005
4. Cimiano, P., *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*. Springer-Verlag New York, Inc.,Secaucus, NJ, USA,2006
5. Sure, Y., Staab, S., et Studer R.: Handbook on Ontologies, chapter 6. On-To-Knowledge Methodology (OTKM). Springer (2003).
6. Maedche, A., Volz. R., The Ontology Extraction and Maintenance Framework Text-To-Onto. *Proceedings of the conference on Data management and Knowledge Management*, DM_KM (2001)
7. Gargouri, Y., Lefebvre, B., Meunier, J.-G.: Ontology Maintenance using Textual Analysis. Proceedings of the Seventh World Multi-Conference on Systemics, Cybernetics and Informatics (SCI). Orlando, USA (2003)
8. Maedche, A., Staab, S.: Measuring Similarity between Ontologies. *Conference on Knowledge Engineering and Management (EKAW 2002),* pp 251–263. Springer-Verlag (2002).
9. Bourigault, D., Fabre, C., Frérot, C., Jacques, M.-P., Ozdowska S. : Syntex, analyseur syntaxique de corpus, in *Actes des 12èmes journées sur le Traitement Automatique des Langues Naturelles*, Dourdan, France. (2005)
10. Van Dyke Parunak, H., Rohwer, R., Belding, T. C., Brueckner, S.: Dynamic decentralized any-time hierarchical clustering. *29th Annual International ACM SIGIR Conference on Research & Development on Information Retrieval*. (2006)
11. Manning C. D. and H. Sch¨utze. *Foundations of Statistical Natural Language Processing.* The MIT Press, Cambridge, Massachusetts, (1999).
12. Di Marzo Serugendo, G., Gleizes, M.-P., and Karageorgos, A. : Self-Organization and Emergence in Multi-Agent Systems. *The Knowledge Engineering Review* Vol.20, N°2, pp165-189 (2005).
13. Ottens, K., Gleizes, M.-P., Glize, P.: A Multi-Agent System for Building Dynamic Ontologies. *International Joint Conference on Autonomous Agents and Multi-agent Systems* (AAMAS 2007). ACM Press, p. 1278-1284 (2007)
14. Assadi, H.,: Construction of a regional ontology from text and its use within a documentary system. *Proceedings of the International Conference on Formal Ontology and Information Systems - FOIS'98* pp236-249. Trento, Italy (1998)
15. Dellschaft, K., Staab, S., On How to Perform a Gold Standard Based Evaluation of Ontology Learning. In *Proc. Of the International Semantic Web Conference 2006*: 228-241 (2006).
16. Brewster, C., Ciravegna., F., Wilks, Y.: Background and Foreground Knowledge in Dynamic Ontology Construction: Viewing Text as Knowledge Maintenance. *Proceedings of the Semantic Web Workshop,* SIGIR (2003).
17. Stojanovic, L., Maedche, A., Motik, B., Stojanovic, N.: User-Driven Ontology Evolution Management. *European Conference on Knowledge Engineering and Management (EKAW 2002*), pp. 285-300. Springer-Verlag (2002).