Un algorithme multi-agent de classification pour la construction d'ontologies dynamiques

Kévin Ottens, Nathalie Aussenac-Gilles

IRIT, Université Paul Sabatier,
118 Route de Narbonne,
F-31062 TOULOUSE CEDEX 9
{ottens,aussenac}@irit.fr,
http://www.irit.fr/

Résumé. La construction d'ontologies à partir de textes reste une tâche coûteuse en temps qui justifie l'émergence de l'*Ontology Learning*. Notre système, *Dynamo*, s'inscrit dans cette mouvance, en apportant une approche originale basée sur une architecture multi-agent adaptative. En particulier, l'article présente le cœur de notre approche, un algorithme distribué de classification hiérarchique qui s'applique sur les résultats d'un analyseur syntaxique. Cet algorithme est évalué et comparé à un algorithme centralisé plus conventionnel. Forts de ces résultats, nous discutons ses limites et dressons en perspective les aménagements à effectuer pour aller vers une solution complète de construction d'ontologies.

1 Introduction

L'analyse syntaxique et terminologique de textes organisés en corpus est envisagée depuis une quinzaine d'années comme une issue possible au problème de la construction d'ontologies ou de terminologies. Non seulement la référence à l'utilisation de la langue est le moyen d'assurer une certaine pertinence aux connaissances ainsi identifiées, mais surtout, la disponibilité de textes sous format électronique autorise le recours à leur analyse par des logiciels de Traitement Automatique des Langues (TAL). Enfin, dans le cas particulier où l'ontologie à construire doit servir à indexer ou annoter des documents par des méta-données, le recours aux textes fournit une terminologie riche et au plus près de celle utilisée dans ces documents.

Plusieurs méthodes ont été définies pour guider ce processus et le rendre systématique, comme Archonte (Bachimont, 2004) ou Terminae (Aussenac-Gilles et al., 2003). Toutefois, ces méthodes se heurtent à la lourdeur du dépouillement des résultats des logiciels d'analyse syntaxique, lexicale ou sémantique des textes auxquels elles ont recours. De plus, toutes s'accordent à souligner le rôle majeur de l'analyste (que nous appelerons aussi par la suite ontologue) dans l'interprétation de ces résultats. Ainsi, lorsqu'un outil d'aide au repérage de relations selon une approche par patron présente des phrases contenant des termes en relation, il n'est pas sûr que cette relation doive être intégrée dans l'ontologie, ou encore qu'elle mette en relation exactement les concepts auxquels renvoient ces termes (Séguéla, 2001).

Aussi, depuis quelques années, plusieurs tentatives sont effectuées pour réduire la charge de l'analyste. Elles profitent des avancées de l'apprentissage automatique ou de méthodes de clas-

sification pour exploiter plus automatiquement les données linguistiques extraites des textes. On parle d'*Ontology Learning* (Maedche, 2002). À titre d'exemple, l'approche préconisée par Maedche et Staab (2000) utilise des heuristiques et un algorithme de fouille de données appliqués aux textes pour « découvrir » des règles d'association entre termes, et définir ainsi à la fois des concepts et des relations entre concepts.

Notre système, Dynamo[1], vise ce même objectif : réduire la part manuelle dans le dépouillement des résultats d'analyse de textes, et suggérer une amorce de réseau conceptuel en vue de construire une ontologie plus efficacement. L'approche retenue est tout à fait originale, et fait appel à un système multi-agent adaptatif. Ces systèmes sont capable de répondre aux changements induits par l'utilisateur en modifiant leur structure. Ce choix est motivé par les qualités qu'offrent les systèmes multi-agents adaptatifs : ils peuvent faciliter la mise au point interactive d'un système (Georgé et al., 2003) (dans notre cas, un réseau conceptuel), permettent sa construction incrémentale automatique par la prise en compte progressive de nouvelles données (issues d'analyses de textes) et enfin, ils peuvent être aisément implémentés de manière répartie. Ainsi le comportement de Dynamo a trois étapes : le système propose une taxonomie, l'utilisateur fait des modifications locales et le système se réorganise pour faire une nouvelle proposition. Contrairement à ASIUM (Faure, 2000), l'utilisateur invervient sur le résultat entier et non sur le contrôle de chaque étape du processus de classification.

Dynamo prend en entrée les résultats d'une analyse syntaxique et terminologique de textes. Il s'appuie sur un algorithme de classification hiérarchique distribué, basé sur un calcul de similarité entre termes qui exploite leurs contextes linguistiques. En sortie, Dynamo présente à l'analyste une organisation hiérarchique de concepts qu'il peut valider, affiner ou corriger, jusqu'à parvenir à un état satisfaisant du réseau sémantique.

Cet article présente l'architecture et l'approche retenue pour le système Dynamo, et particulièrement l'algorithme de classification hiérarchique distribuée implémenté dans les agents (section 2.2). Après une étude quantitative et qualitative des performances de cet algorithme (section 3), nous soulignons l'apport d'une approche multi-agent pour la classification. Enfin, nous discutons de son utilisation dans la perspective de construire des ontologies (section 4).

2 Présentation du système Dynamo

2.1 Architecture proposée

Dans cette section, nous présentons l'architecture de Dynamo. Elle se situe dans la lignée d'une réflexion portée dans un contexte plus global de gestion et de maintenance d'ontologies dynamiques en lien avec des collections de documents (Ottens et al., 2004). Contrairement à la réflexion précédente qui cherchait, entre autre, à aborder les besoins du Traitement Automatique des Langues (TAL), ici, nous nous attachons uniquement à combler les besoins de l'Ingénierie des Connaissances (IC).

Le système Dynamo se compose de trois grandes parties (cf. figure 1) :

– le réseau de termes, obtenu en sortie d'un extracteur de termes utilisé pour pré-traiter le corpus textuel ; il s'agit de syntagme nominaux sous forme lemmatisée comme par exemple « hépatite viral » ;

[1] Dynamic Ontologies

– le système multi-agent qui utilise le réseau de termes pour effectuer une classification hiérarchique afin d'obtenir une taxonomie ; les agents du système coopèrent afin se se positionner dans une hierarchie, le système multi-agent constitue la taxonomie ;
– l'interface permettant à l'utilisateur de visualiser et contrôler le processus de classification et ses résultats.

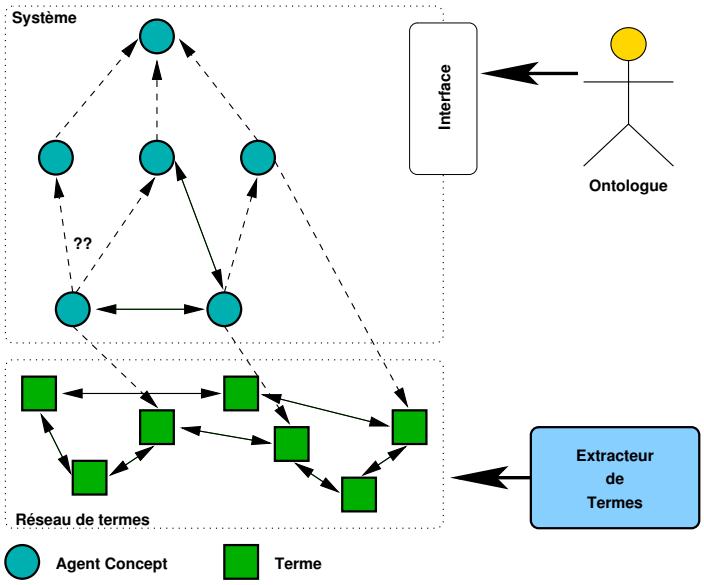

FIG. 1 – *Architecture du système*

L'extracteur de termes choisi est le logiciel Syntex, utilisé notamment pour des tâches de construction d'ontologies (Bourigault et Aussenac-Gilles, 2003). Nous l'avons sélectionné principalement pour sa robustesse et la grande quantité d'informations extraites. En particulier, le réseau « Tête-Expansion », créé par cet outil, a déjà prouvé être une structure intéressante pour un système de classification (Assadi, 1998). Dans un tel réseau, chaque terme est relié, d'une part à sa tête[2] et son expansion[3], et d'autre part à tous les termes dont il est lui-même tête ou expansion. Par exemple, « algorithme centralisé de classification » a comme tête « algorithme centralisé » et comme expansion « classification ». De même, « algorithme centralisé » est composé de « algorithme » et « centralisé ».

Le réseau de termes récupéré en sortie de l'extracteur est stocké dans une base de données. Pour chaque paire de termes, on suppose qu'il est possible de calculer son indice de similitude (ou similarité), en vue d'effectuer une classification (Faure, 2000) (Assadi, 1998). De par la nature des données, nous nous intéressons uniquement à des indices de similarité entre objets décrits par des variables binaires, c'est-à-dire qu'un individu est décrit par la présence ou l'absence d'un ensemble de caractéristiques (Saporta, 1990). Dans le cas de termes, il s'agit, en général, des contextes d'utilisation. Avec Syntex, ces contextes sont identifiés par des termes et des relations syntaxiques caractérisées.

[2] *i.e.* le sous-syntagme maximal situé en tête du terme
[3] *i.e.* le sous-syntagme maximal situé en queue du terme

Le système multi-agent implémente l'algorithme de classification distribué décrit en détail dans la section 2.2. Il est conçu pour être à la fois le système produisant la structure résultante et la structure elle-même. En ce sens, chaque agent est une entité informatique représentant un concept, dont le comportement autonome lui permet de trouver sa place dans l'organisation, à savoir l'ontologie. Ils disposent tous de capacités de communication et d'algorithmes leur permettant de s'approcher ou s'éloigner selon différents critères. La sortie du système est donc l'organisation obtenue par l'interaction des agents, tout en tenant compte des pressions extérieures exercées par l'ontologue lorsqu'il effectue des modifications de la taxonomie selon ses besoins. Dans cet article, nous nous intéressons principalement au processus de classification distribué dans les agents, bien que la création d'une ontologie ne se réduise pas à cet aspect.

2.2 Un algorithme de classification distribué

Cette section présente l'algorithme de classification distribué, utilisé dans le système. Afin d'en améliorer la compréhension, et en vue de son évaluation dans la section 3, nous rappelons ci-dessous l'algorithme de base utilisé pour une classification hiérarchique ascendante dans un espace non métrique, mais dont la mesure de similitude utilisée est symétrique (Saporta, 1990) (ce qui est le cas avec les mesures utilisées dans notre système).

Algorithme 1 : Algorithme centralisé de classification hiérarchique ascendante

```
Entrées : La liste L des individus à hiérarchiser
Sorties : La racine R de l'arbre de classification
tant que longueur(L) > 1 faire
    max ← 0;
    A ← nil;
    B ← nil;
    pour i ← 0 à longueur(L) faire
        I ← L[i];
        pour j ← i + 1 à longueur(L) faire
            J ← L[j];
            sim ← similitude(I, J);
            si sim > max alors
                max ← sim;
                A ← I;
                B ← J;
            fin
        fin
    fin
    retirer(A, L);
    retirer(B, L);
    ajouter((A, B), L);
fin
R ← L[0];
```

Dans l'algorithme 1, à chaque étape de la classification, la paire des éléments les plus similaires est déterminée. Ces deux éléments sont regroupés, et la classe résultante est insérée dans la liste des éléments restants. L'algorithme s'arrête quand cette liste ne contient plus qu'un élément.

La structure hiérarchique résultante de l'algorithme 1 est nécessairement un arbre binaire de par le regroupement deux à deux. Le regroupement des éléments les plus similaires revient aussi à les éloigner des éléments dont ils sont le plus dissimilaires. L'algorithme distribué multi-agent présenté est conçu autour de ces deux constats. Cet algorithme est exécuté de manière concurrente par chacun des agents du système.

Notons aussi que dans la suite de l'article nous utilisons lors de l'évaluation des deux algorithmes un stratégie d'agrégation en moyenne (Saporta, 1990) et l'indice de similarité utilisé sur des termes dans Assadi (1998) ayant la formule suivante pour deux termes i et j :

$$sim(i,j) = \frac{\alpha}{2}(\frac{a}{a+b} + \frac{a}{a+c}) + \frac{1-\alpha}{2}(\frac{d}{d+c} + \frac{d}{d+b})$$

Où, a, b, c et d sont respectivement le nombre de contextes partagés par i et j, présents uniquement chez i, présents uniquement chez j, et présents ni chez i ni chez j. Les contextes sont déterminés en explorant le réseau « Tête-Expansion » (Assadi, 1998). Pour nos tests, nous avons fixé α à $0,75$. Tous ces choix conditionnent l'arbre résultat, mais n'influencent ni le déroulement général de l'algorithme ni sa complexité.

Le système est initialisé de la manière suivante :

– un agent *TOP* n'ayant aucun parent est créé, il sera la racine de la taxonomie résultante ;
– un agent est créé pour chacun des concepts à positionner dans la taxonomie, ils ont tous *TOP* comme parent ; ces concepts initiaux ont tous un terme du réseau comme référent.

Une fois cette structure de base en place, l'algorithme se déroule en parallèle au sein de chaque agent, jusqu'à obtenir une position d'équilibre global et donc une première version de la taxonomie résultante est présentée à l'utilisateur. Par la suite, les modifications effectuées par l'utilisateur dans la taxonomie auront pour effet de réactiver les agents concernés, l'agorithme sera donc relancé localement.

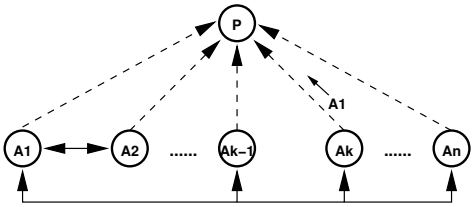

FIG. 2 – *Classification distribuée : Etape 1*

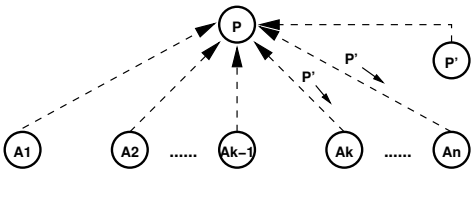

FIG. 3 – *Classification distribuée : Etape 2*

La première étape du procédé (figure 2) se déroule en parallèle dans les agents (ici on s'intéressera uniquement à A_k) ayant plus d'un frère (puisque nous cherchons à obtenir un arbre binaire). L'agent A_k envoie alors un message à son père P indiquant le frère dont il est le plus dissimilaire (ici A_1). P reçoit donc un message du même type de chacun de ses fils. Par la suite, ce type de message sera appelé un « vote ».

Ensuite, lorsque P a reçu les messages de tous ses fils, il exécute la deuxième étape (figure 3). Grâce aux messages reçus indiquant les préférences de ses fils, P peut constituer trois sous-groupes parmi ses fils :

– le fils qui a reçu le plus de « votes » par ses frères, c'est-à-dire le fils étant le plus dissimilaire du plus grand nombre de ses frères. En cas d'égalité, un des *ex aequo* est choisi au hasard (ici A_1) ;

– les fils ayant permis « l'élection » du premier groupe, c'est-à-dire les agents ayant choisi leur frère du premier groupe comme étant celui qui leur est le plus dissimilaire (ici A_k à A_n) ;
– les fils restants (ici A_2 à A_{k-1}).

P crée alors un nouvel agent P' (de père P) et demande aux agents du second groupe (ici les agents A_k à A_n) d'en faire leur nouveau père.

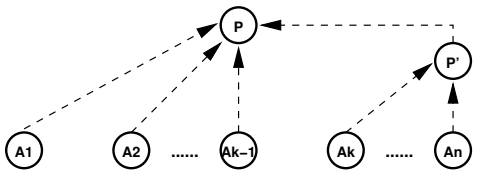

FIG. 4 – *Classification distribuée : Etape 3*

Enfin, l'étape 3 (figure 4) est assez évidente. Les fils repoussés par P (ici les agents A_k à A_n) tiennent compte de son message et choisissent P' comme nouveau père. La hiérarchie se constitue ainsi de proche en proche.

Remarquons que cet algorithme converge nécessairement, puisque le nombre de frères d'un agent va en diminuant. Lorsqu'un agent n'a plus qu'un seul frère, son activité est stoppée (hors traitement des messages de ses fils). Il s'agit donc bien d'un algorithme de classification hiérarchique distribué, les décisions étant prises par des négociations au sein de groupes d'agents autonomes. Les traitements ainsi que les connaissances étant locales aux agents, et la communication se faisant par envoi de message, un tel algorithme peut tout à fait être réparti sur un réseau de machines.

3 Évaluation des performances

3.1 Approche quantitative

À présent, nous allons évaluer les propriétés de notre algorithme distribué. Pour cela, il convient de commencer par une évaluation quantitative, basée sur sa complexité, tout en le comparant à l'algorithme 1 de la section précédente.

La complexité théorique sera calculée pour le pire cas, en considérant l'opération de calcul de la similitude entre deux éléments comme élémentaire. Pour l'algorithme distribué, le pire cas signifie qu'à chaque étape, on ne peut constituer qu'un seul groupe de deux individus. Dans ces conditions, pour un jeu de données initial de n éléments, les deux algorithmes donnent le nombre de calculs de similitude suivant[4] :

$$t_1(n) = \sum_{i=1}^{n} \frac{i \times (i-1)}{2} \qquad t_{dist}(n) = \sum_{i=1}^{n} i \times (i-1)$$

[4]Les calculs menant à ces équations ne sont pas donnés par souci de concision

Ici, $t_1(n)$ et $t_{dist}(n)$ sont le nombre de calculs de similitudes effectués respectivement par l'algorithme centralisé et par l'algorithme distribué sur l'ensemble du système. Les deux algorithmes ont donc une complexité en $O(n^3)$. Dans le pire cas, l'algorithme distribué effectue deux fois plus d'opérations élémentaires que l'algorithme centralisé. Cette différence de facteur 2 provient de la localité des prises de décision au sein des agents. Ainsi, les calculs de similitudes sont faits deux fois pour chaque paire d'agents. On pourrait envisager qu'un agent, après avoir effectué un tel calcul, envoie son résultat à l'autre partie. Toutefois, cela reviendrait simplement à déplacer le problème en générant plus de communications au sein du système.

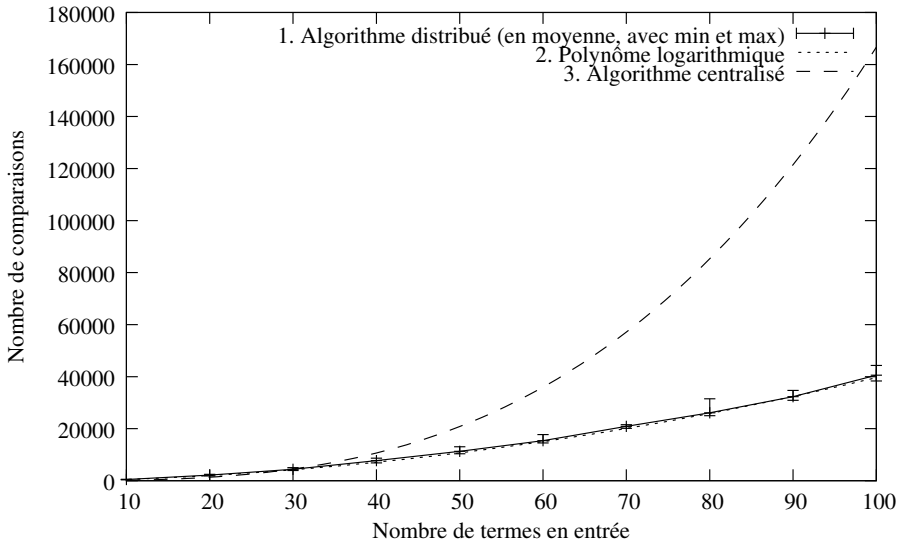

FIG. 5 – *Résultats expérimentaux*

La complexité en moyenne de l'algorithme a été déterminée expérimentalement. Pour cela, le système multi-agent a été exécuté avec des jeux de données en entrée allant de dix à cent termes. La valeur retenue est la moyenne du nombre de comparaisons effectuées pour une centaine d'exécutions sans intervention de l'utilisateur. Ceci donne les courbes de la figure 5. L'algorithme distribué est donc en moyenne plus efficace que l'algorithme centralisé, et sa complexité en moyenne plus réduite que pour le pire cas. Cela s'explique simplement par la faible probabilité qu'un jeu de données pousse le système à ne constituer que des groupes minimaux (deux éléments) ou maximaux ($n-1$ éléments) à chaque étape du raisonnement. La courbe 2 représente le polynôme logarithmique minimisant l'erreur avec la courbe 1. Le terme de plus fort degré de ce polynôme est en $n^2log(n)$, donc notre algorithme distribué a en moyenne une complexité en $O(n^2log(n))$. Enfin, on remarque l'écart réduit entre les performances en moyenne, au maximum et au minimum. Dans le pire cas, pour 100 termes, l'écart type est de 1960,75 pour une moyenne de 40550,70 (soit environ 5%) ce qui souligne la stabilité du système.

3.2 Constat qualitatif

Bien que les résultats quantitatifs constatés soient intéressants, le réel intérêt de l'approche relève de traits plus qualitatifs que nous présentons dans cette section. Tous sont des avantages obtenus grâce à l'utilisation d'une approche multi-agent.

Le principal avantage de l'utilisation d'un système multi-agent adaptatif pour une tâche de classification est le côté dynamique introduit par un tel système. L'utilisateur peut intervenir pour effectuer des corrections et la hiérarchie s'adapte en fonction de cette demande. C'est particulièrement intéressant dans un contexte d'IC. En effet, la hiérarchisation fournie par le système est amenée à être modifiée par l'utilisateur puisqu'elle est le résultat d'un traitement statistique. Lors de retours nécessaires au texte pour examiner les contextes d'usage des termes (Aussenac-Gilles et Condamines, 2004), l'ontologue pourra interpréter le contenu réel et ainsi réviser la proposition du système. Ceci est très difficile à réaliser avec une approche centralisée. Dans la plupart des cas, il faut trouver l'étape du raisonnement qui a engendré le résultat erroné et modifier la classe correspondante manuellement. Malheureusement, dans ce cas, toutes les étapes de raisonnement subséquentes à la création de la classe modifiée sont perdues et doivent être recalculées en tenant compte de la modification. C'est pourquoi un système comme ASIUM (Faure, 2000) présente à l'utilisateur les classes créées à chaque étape du raisonnement, pour essayer de gommer ce problème grâce à une collaboration système-utilisateur. De plus, la hierarchie complète n'est visible qu'à la fin du processus. Ainsi, l'utilisateur peut ne constater l'introduction d'une erreur que trop tard. Dans Dynamo, on laisse l'algorithme se dérouler, bien que l'utilisateur ait l'opportunité de l'interrompre et le relancer quand il le souhaite. Il dispose à tout instant d'une hiérarchie complète qu'il peut modifier. Ses interventions déclencheront l'algorithme localement jusqu'à obtenir un nouvel état stable.

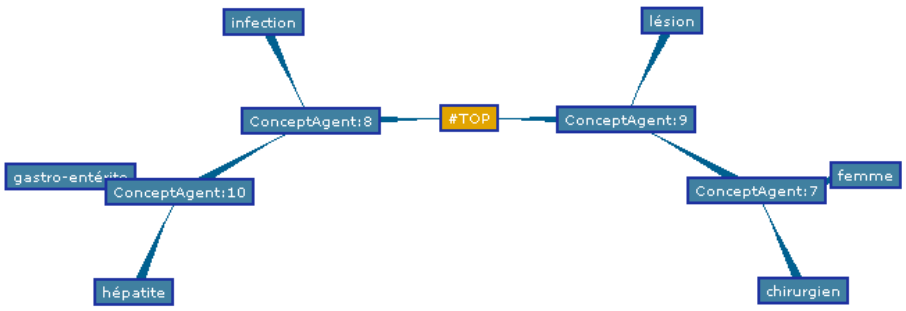

FIG. 6 – *Arbre d'agents concept après stabilisation autonome du système*

Afin d'illustrer notre propos, nous présentons un exemple simple à l'aide de captures d'écran du système en fonctionnement. En utilisant un jeu de test et en laissant le système travailler de lui-même, on obtient l'arborescence de la figure 6 après stabilisation[5]. Il est clair que le concept représenté par le terme « lésion » est mal placé. Il se trouve que les calculs de similitudes le placent plus proche de « femme » et « chirurgien » que d'« infection », « gastro-entérite » et « hépatite ». Cette mauvaise position pour « lésion » s'explique alors par le fait

[5]Dans la suite de l'article, toutes les figures d'arbres sont obtenues par captures d'écrans du système en fonctionnement

que sans intervention de l'ontologue, le système effectue le raisonnement uniquement sur ce critère statistique.

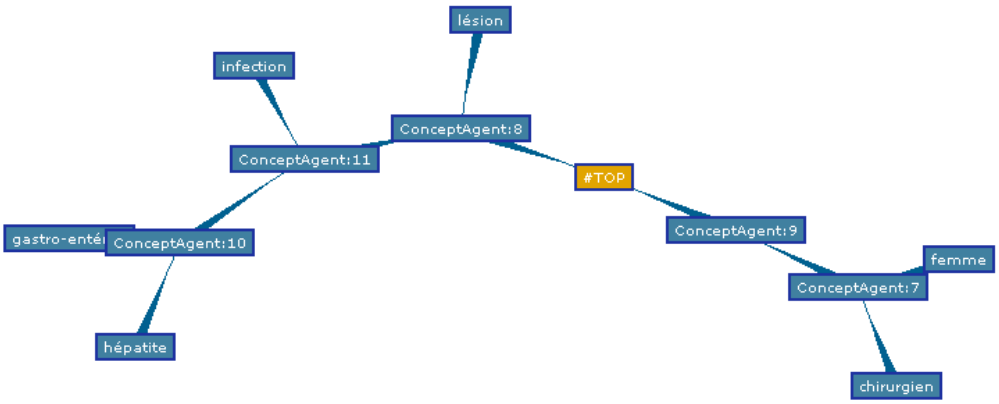

FIG. 7 – *Arbre d'agents concept après intervention de l'utilisateur*

L'utilisateur intervient alors pour le replacer dans la bonne branche, en lui affectant « ConceptAgent :8 »[6] comme nouveau père. Le système réagit ensuite de lui-même pour raffiner l'arbre de classification et revenir à un arbre binaire, créant donc « ConceptAgent :11 ». Le nouvel état stable est alors celui de la figure 7.

Ce couplage système-ontologue est indispensable pour une création d'ontologie. Pour le mettre en place, aucun ajustement particulier sur le principe de l'algorithme distribué n'a été nécessaire car chaque agent effectue un traitement local autonome.

De plus, cet algorithme peut être réparti sur un réseau *de facto*. Il est tout à fait envisageable d'exécuter les agents sur des machines différentes. La communication entre les agents se fait par envoi de messages et chacun d'entre eux conserve son autonomie de décision. Donc, une telle modification du système, pour lui permettre de tourner sur plusieurs machines, ne nécessiterait aucun ajustement de l'algorithme. En revanche, cela demanderait de revoir la couche de communication et la gestion de la création des agents dans notre implémentation actuelle.

4 Discussion & Perspectives

4.1 Limites actuelles du modèle

Comme constaté dans la section 3, les gains en complexité de notre approche sont plutôt faibles ($O(n^2 log(n))$ en place de $O(n^3)$), mais cela signifie un nombre de traitements réduit (comparativement à un algorithme centralisé classique) lorsque le nombre de concepts dans l'ontologie devient suffisamment grand. Toutefois, il convient de remarquer que l'usage même d'un système multi-agent consomme nécessairement plus de ressources qu'un algorithme cen-

[6] « ConceptAgent :X » est le nom automatiquement donné à un agent concept qui n'est pas directement représenté par un terme.

tralisé. En effet, le travail de synchronisation s'effectue grâce à des envois de message, un tel système est donc plus coûteux en temps machine qu'une implémentation centralisée.

Le principal point faible actuel de notre algorithme est que le résultat est dépendant de l'ordre d'ajout des données. Lorsque le système travaille de lui-même sur un jeu de données fixe, fourni à l'initialisation, le résultat final est équivalent à ce qui peut être obtenu avec un algorithme centralisé. En revanche, l'ajout d'un nouvel élément après une première stabilitisation a un impact sur le résultat final.

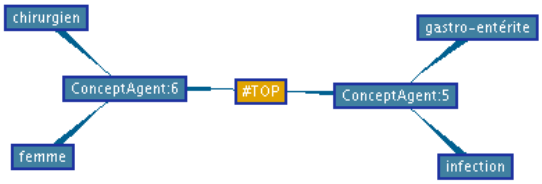

FIG. 8 – *Arbre d'agents concept après stabilisation autonome du système*

Afin d'illustrer notre propos, nous présentons un autre exemple de fonctionnement du système. En utilisant un jeu de test et en laissant le système travailler de lui-même, on obtient l'arborescence de la figure 8 après stabilisation.

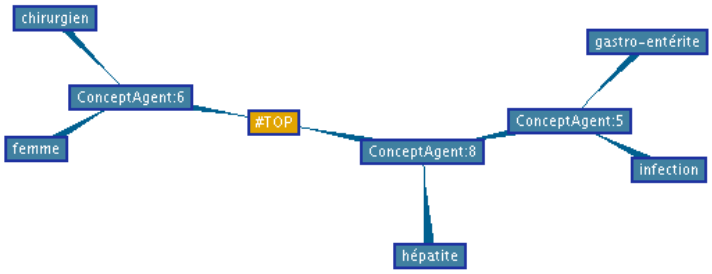

FIG. 9 – *Arbre d'agents concept après prise en compte de « hépatite »*

L'utilisateur intervient alors et ajoute un nouveau concept représenté par le terme « hépatite » sous la racine. Le système réagit et se stabilise, on obtient en résultat l'organisation présentée en figure 9. « hépatite » est bien situé dans la bonne branche, mais on ne retrouve pas la configuration désirée de la figure 6 du précédent exemple. Pour cela, il faut compléter notre algorithme distribué pour permettre à un concept de se déplacer le long d'une branche. Nous travaillons actuellement sur ces règles, mais la comparaison avec des algorithmes centralisés deviendra alors plus difficile.

4.2 Vers la construction d'ontologies

Notre projet n'a pas pour finalité de proposer un algorithme distribué de classification hiérarchique, auquel cas celui présenté, qui crée un arbre binaire, suffirait, mais d'obtenir une

taxonomie dynamique. Cela signifie que le système présenté dans cet article doit encore évoluer afin d'être utilisable pour cette tâche.

Tout d'abord, dans le cas général, faute d'un voisinage utilisable dans le réseau « Tête-Expansion », certains termes extraits par Syntex ne sont pas traitables en utilisant notre algorithme de classification hiérarchique. Pourtant ils seraient intéressants pour la taxonomie. Il est donc nécessaire d'ajouter des capacités à nos agents pour tenir compte d'autres critères pour se positionner. En particulier, les éléments de plus bas niveau de la taxonomie pourraient être positionnés en examinant aussi la relation « Tête » des termes qui les représentent. Il a été constaté qu'il s'agissait d'un travail effectué par les utilisateurs de Syntex pour la structuration des niveaux les plus bas de la taxonomie (Aussenac-Gilles et al., 2003). Cela permettrait de rapprocher par exemple « infection » et « infection viral » qui, dans notre corpus de test, ne pourraient être rapprochés par le système actuel que si leurs contextes d'utilisation étaient identiques.

Une ontologie n'est que rarement un arbre binaire, il faudra donc ajouter des critères supplémentaires afin d'effectuer des regroupements et ainsi obtenir des noeuds n-aires. De la même manière, les interventions de l'utilisateur poussent le système à introduire des niveaux dans la hiérarchie qui ne sont pas forcément nécessaires. On obtient parfois des situations dégradées où un noeud n'a qu'un seul fils. Il s'agit donc des deux axes de simplification à explorer pour obtenir une structure plus raffinée.

5 Conclusion

Après avoir été présentée comme une solution prometteuse, assurant la qualité des modèles et leur richesse terminologique, la construction d'ontologies à partir de l'analyse de corpus s'avère longue et coûteuse. Elle requiert la supervision de l'analyste et la prise en compte de l'objectif d'utilisation de l'ontologie. L'utilisation de logiciels de TAL facilite l'étude des connaissances repérables dans les textes à travers l'utilisation du langage. Cependant, ces logiciels produisent de gros volumes d'informations lexicales ou grammaticales qu'il n'est pas simple d'exploiter pour définir des éléments conceptuels. Notre contribution se situe à cette étape du processus de modélisation à partir de textes, avant toute forme de normalisation ou différenciation.

Nous proposons une approche de classification hiérarchique distribuée, implémentée à l'aide d'un système multi-agent adaptatif, pour suggérer à l'analyste une première structure taxonomique organisant des concepts. Notre système exploite en entrée le réseau terminologique résultant d'une analyse syntaxique de corpus faite par Syntex. L'état courant du développement permet de produire des structures simples, de les soumettre à l'analyste et de les faire évoluer en fonction des corrections qu'il a apportées. Les performances de cet algorithme distribué sont comparables à celle de la version centralisée. Ses points forts sont essentiellement qualitatifs, car il autorise des interactions avec l'utilisateur et une adaptation progressive à l'ajout de nouveaux éléments linguistiques. La décentralisation permet également de répartir les calculs effectués.

Dans la perspective de construire une ontologie, ce travail n'est qu'une ébauche. Il doit être poursuivi, à la fois pour assurer une meilleure robustesse à la classification, et pour parvenir à des structures plus riches que de simples arbres. Parmi ces enrichissements, un point difficile sera certainement l'étiquetage des relations et la gestion différenciée des types de relation.

Références

Assadi, H. (1998). *Construction d'ontologies régionales à partir de textes techniques*. Thèse de doctorat, Université Paris VI.

Aussenac-Gilles, N., B. Biébow, et S. Szulman (2003). D'une méthode à un guide pratique de modélisation de connaissances à partir de texte. *TIA 2003*.

Aussenac-Gilles, N. et A. Condamines (2004). Documents électroniques et constitution de ressources terminologiques ou ontologiques. *Revue I3 4*(1), 75–94.

Bachimont, B. (2004). *Art et sciences du numérique : ingénierie des connaissances et critique de la raison computationnelle.* Mémoire d'habilitation à diriger des recherches, Université Technologique de Compiègne.

Bourigault, D. et N. Aussenac-Gilles (2003). Construction d'ontologies à partir de textes. *10ème conf. sur le Traitement Automatique des Langues Naturelles TALN2003*, 27–47.

Faure, D. (2000). *Conception de méthode d'apprentissage symbolique et automatique pour l'acquisition de cadres de sous-catégorisation de verbes et de connaissances sémantiques à partir de textes : le système ASIUM*. Thèse de doctorat, Université Paris XI Orsay.

Georgé, J.-P., G. Picard, M.-P. Gleizes, et P. Glize (2003). Living Design for Open Computational Systems. *12th IEEE International Workshops on Enabling Technologies, Infrastructure for Collaborative Enterprises*, 389–394.

Maedche, A. (2002). *Ontology learning for the Semantic Web*. Kluwer Academic Publisher.

Maedche, A. et S. Staab (2000). Mining Ontologies from Text. *EKAW 2000*, 189–202.

Ottens, K., N. Aussenac-Gilles, M.-P. Gleizes, et P. Glize (2004). Systèmes multi-agents pour l'extraction d'ontologies à partir de textes : revue de questions. *AGENTAL : Agents et Langue, Journée d'étude ATALA sur les Relations entre Systèmes Multi-Agents et Traitement Automatique des Langues*.

Saporta, G. (1990). *Probabilités Analyse des Données et Statistique*. Technip.

Séguéla, P. (2001). *Construction de modèles de connaissances par analyse linguistique de relations lexicales dans les documents techniques*. Thèse de doctorat, Univ. Toulouse III.

Summary

Ontologies building from text is still a time-consuming task which justifies the growth of Ontology Learning. Our system named *Dynamo* is designed along this philosophy but following an original approach based on a multi-agent architecture. In particular, a distributed hierarchical clustering algorithm, core of our approach, is evaluated and compared to a more conventional centralized algorithm. With those results in mind, we discuss its limits and add as perspectives the modifications required to reach a complete ontology building solution.