Plate-forme de veille multi-agents pour l'aide à la décision

Réalisation d'un système de veille basé sur une approche multi agents

Nicolas Chanchevrier* — Xavier Denis*,**

* EADS S&DE Division TI
Parc d'Affaires des Portes - BP613 27106 Val de Reuil Cedex
nchanchevrier@matra-ms2i.fr, xavier.denis@tiscali.fr
** Laboratoire d'Informatique du Havre
25 rue Philippe Lebon - BP 1123 76063 Le Havre Cedex

Résumé. L'essor d'Internet ces dix dernières années a favorisé l'apparition de nombreuses informations disponibles en ligne, trouvant une utilité dans la veille sur les domaines les plus divers tels que la veille stratégique & concurrentielle, la veille technologique. Cependant, la quantité d'information publiée est telle qu'il est humainement impossible de prendre connaissance de l'ensemble des données. Ainsi, nombre de recherches éparses ont vu le jour pour traiter des problématiques ciblées de fouille de texte, mais peu de systèmes de veille automatisée sont actuellement mis en œuvre, permettant l'interopérabilité de différentes approches d'analyse de contenu. La plateforme qui sera réalisée, dont une présentation succincte est donnée, repose sur un système multi agent où chaque entité joue un rôle dans le processus de veille (acquisition des données, analyse du contenu, diffusion de l'information, apprentissage), la partie innovante majeure provenant de la gestion par l'utilisateur des réseaux d'accointance des agents.

1. Introduction

La quantité d'information disponible sur Internet est en constante augmentation et la nécessité de pouvoir surveiller des contenus ciblés est devenue une priorité. Malheureusement cette tâche est irréalisable sans l'utilisation active des moyens d'analyse automatiques et donc des technologies de textmining. Un certain nombre d'efforts ont été réalisés pour développer des outils capables de surveiller des sites (Coppernic Agent, WebWatcher) mais aucun ne dispose d'une architecture suffisamment malléable pour effectuer des traitements plus fins. L'approche qui a été retenue se base sur un système multi agent où chaque agent à un but (récupérer l'information, l'analyser, alerter, etc.) et où les communications sont dirigées par l'utilisateur. Il s'agit donc de mettre à la disposition de veilleurs un outils ouvert proposant des briques de base et permettant l'intégration et l'interopérabilité avec d'autres fonctions utiles dans le contexte de la veille sur Internet.

2. Description formelle d'un système de veille

2.1. Introduction

Ce paragraphe est destiné à définir ce qu'est exactement la notion de veille et les étapes nécessaires pour que le processus puisse fonctionner selon un procédé déterminé. En particulier il a pour but de spécifier quels sont les utilisateurs d'un système de veille et quelle est leur participation dans la réalisation d'une démarche de surveillance.

2.2. La démarche de veille

La démarche de veille peut être considérée comme un processus constant décomposé en deux étapes majeurs : La surveillance et l'exploitation [Goujon B, 2000]

2.2.1. La surveillance

C'est la phase qui consiste à trouver les informations nécessaires, à les récupérer et à les diffuser (ou partager). Cette première étape soulève un certain nombre de problèmes pour lesquels des solutions doivent être apportées : elle sont proposées ci-après.

2.2.1.1. Localisation des données

Comme expliqué dans l'introduction, Internet regorge de nombreux documents mais il est souvent difficile de pouvoir les localiser très précisément, en particulier à cause du fonctionnement même des URL. En effet, il n'est pas rare de trouver à la place d'un document souhaité un message nommé « Error 404 » qui signifie que le lien spécifié n'est plus valide. Ce problème concerne bien sûr les pages Web mais se pose aussi pour toutes les ressources volatiles telles que les mails, les fichiers sur un disque dur ou les canaux de discussion. Or la veille nécessite de posséder un accès définitif à ces fichiers dans le but de comparaison et de diffusion. Le stockage en masse des données est donc quasiment inévitable. La légalité des sources est aussi un problème puisque les documents dits « sensibles » se trouvent souvent sur des ordinateurs pirates ou des endroits de stockage protégés (mot de passe, accès payant, etc.).

2.2.1.2. Collecte des données

Comme indiqué précédemment, les systèmes de veille devront inévitablement conserver une trace des informations manipulées. Cela pose le problème de la représentation de l'information et de son stockage. En effet, les sources de données sont souvent de nature hétérogène car elles concernent aussi bien les pages Web (données structurées) que les mails (données semi structurées) ou que les documents textuels (données non structurées). De plus, le contenu à analyser ne concernera pas forcément l'intégralité du document puisque, pour l'exemple des pages Web, il est seulement utile de conserver la partie textuelle de la page (ce qu'il y a d'affiché dans le navigateur) et de s'affranchir des balises et autre données qui servent à structurer le texte (C'est évident pour les fichiers Web HTML, mais un peu moins pour les fichiers Word ou PDF…). Cela amène donc à la définition d'un objet de stockage textuel générique qui va permettre de garder les données textuelles tout en conservant la structure du document original (Figure 1, en page 3) [Denis X, 2001] :

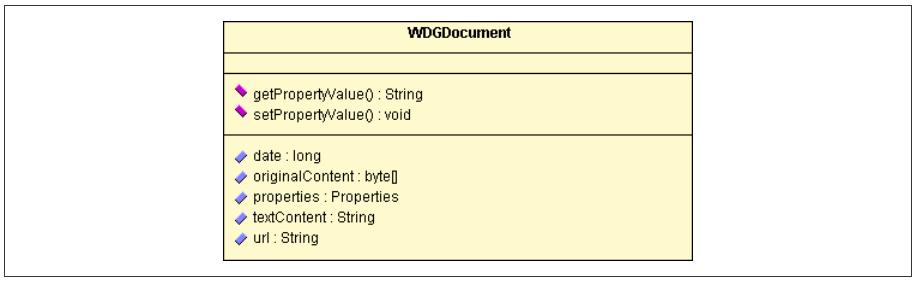

Figure 1. Diagramme UML de la classe de stockage des documents

On retrouve dans ce diagramme le contenu original du document (sous forme d'un byte[] pour le stockage binaire des données) et le contenu textuel (sous forme de String). A ce document sont aussi rattachées des propriétés qui seront définies au fur et à mesure des traitements effectués.

2.2.1.3. Diffusion de l'information

L'information ayant été récupérée, il s'agit maintenant de l'envoyer au bon endroit[1] pour que le processus de veille puisse se poursuivre. De nos jours, cette diffusion est quasiment toujours effectuée de manière électronique par envoi de messages informatisés. Associée aux deux étapes précédentes, cela permet d'initialiser un traitement informatique des données.

2.2.2. L'exploitation

L'exploitation concerne le traitement automatisé des informations recueillies. Elle se décompose en trois étapes principales.

2.2.2.1. Le traitement

Les documents téléchargés contiennent souvent des informations parasites qui pourraient induire des erreurs lors des traitements automatisés suivants. Le but de cette étape est justement d'épurer les données (clés PGP, signatures, …) pour ne garder que la partie vraiment utile [Yang Y, 1995)]

2.2.2.2. L'analyse et la validation

C'est la partie la plus importante pour le veilleur puisque c'est elle qui va fournir les résultats recherchés. Elle est en général composée d'outils de filtrage de documents (par mot clé ou autre), de clusterisation (regroupement de textes par thèmes identiques) et autres analyseurs qui peuvent aider le veilleur à synthétiser les documents reçus (analyse linguistique par exemple).

La validation concerne plus particulièrement les indications que vont fournir les veilleurs concernant le bon ou mauvais positionnement des documents par rapport aux filtres ou aux outils d'analyse.

1 Personne physique ou bien traitement informatique.

2.2.2.3. L'utilisation

Dernière étape du processus qui consiste à anticiper l'évolution des marchés à l'aide des indices fournis par les outils de veille.

Ces six étapes sont résumées dans la *Figure 2 en page 5*.

2.3. Les différents protagonistes de la veille

Ils sont au nombre de trois et ont chacun un rôle bien précis qui sont décrits dans les chapitres suivants :

2.3.1. Les administrateurs (veilleurs)

Ils ont en charge la localisation des sources de données (appelés « pointeurs utiles ») et interviennent donc en amont du processus de veille (rapatriement appelé « dump »). De la qualité des sources fournies dépend la qualité du résultat. En fait ce sont eux qui possèdent la connaissance technique et pratique des sites où se trouvent les informations recherchées car ils possèdent en général des bases de données complètes sur des références classées par thème. Les veilleurs interviennent aussi dans la configuration de l'outil de veille (voir chapitre suivant concernant la gestion des agents).

2.3.2. Les utilisateurs finaux

Ce sont eux qui vont recevoir les alertes délivrées par le système de veille. Ils ne sont pas sensés connaître la façon dont est configuré l'outil mais peuvent intervenir indirectement sur celui-ci par un mécanisme appelé « retour d'expérience utilisateur ».

2.3.3. Les agents

Les agents sont des entités informatiques autonomes et communicantes qui sont présentes dans le système pour aider l'utilisateur à obtenir les résultats qu'il souhaite. Dans notre système, ils interviennent dans toutes les étapes du processus de veille et font l'interface entre les deux protagonistes cités précédemment et le logiciel lui-même. Une description plus formelle de leur fonctionnement est donnée dans les chapitres suivants.

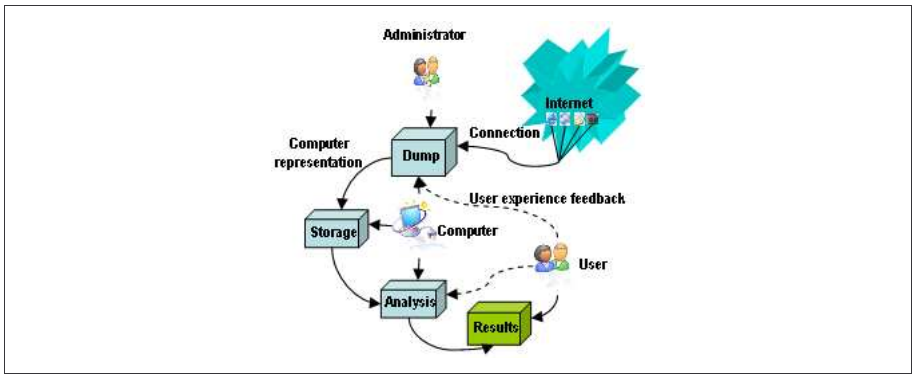

Figure 2. Schéma général de fonctionnement d'une veille

2.4. Les outils existants

Il existe un certain nombre d'outils qui permettent de faire de la veille (MyUpdate, NetVigie, TeleportPro [VEI02]), ou tout du moins un certain nombre des six étapes définies précédemment. La plupart des outils se content en général de collecter l'information (en la filtrant par mot-clés) et la présentent à l'utilisateur final. Cependant, très peu savent prendre en compte les critiques de l'utilisateur pour influencer les résultats fournis. L'approche qui est proposée dans ce papier consiste à utiliser des agents et à les faire fonctionner de manière collaborative pour obtenir des fonctionnalités et des résultats que les autres outils n'atteignent pas.

3. Utilisation des agents pour la veille

Ce chapitre est dédié à la description des agents qui sont disponibles dans le système de veille proposé et en particulier la façon dont les protagonistes humains (administrateurs et utilisateurs) vont interagir avec eux.

3.1. Principe de flux d'analyse

Comme indiqué dans le chapitre d'introduction, la veille est un processus d'analyse continu. Une des façons de simuler cette action est de créer un flux *(Figure 3 en page 6*) dans laquelle les agents vont représenter les nœuds et les communications serviront de lien entre ces nœuds (principe de l'outil de datamining Clementine [Goebel et Le Gruenwald,1999]). Il est à noter que les messages conserveront une trace des agents par lesquels ils sont passés de manière à pouvoir identifier la chaîne empruntée par un document (voir chapitre sur le retour d'expérience utilisateur).

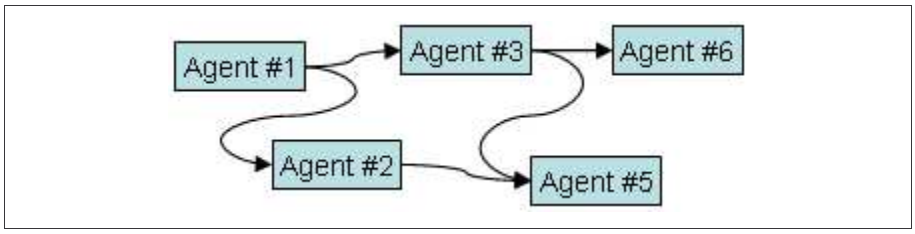

Figure 3. **Enchaînement des agents**

Pour réaliser une veille, un administrateur devra donc :
– Sélectionner les agents à utiliser
– Les relier pour créer le flux (macro configuration)
– Les configurer un par un (micro configuration)

3.1.1. Les agents

Les agents peuvent être considérés dans la plateforme comme des outils indépendants qui possèdent une tâche principale (but) [Ferber J, 1995]. Le flux que l'on peut créer implique que les agents possèdent un certain nombre d'entrées et de sorties qui servent pour les communications. Leur algorithme de fonctionnement est donné *Figure 4 ci-dessous*.

```
Tant Que agent_vivant()
  Si agent_a_un_timing() et timing_valide()
    executer(messages_en_attente())
  Sinon
    Si agent_en_pause == faux
      Si executer_une_fois()
        executer(premier_message_en_attente())
      Sinon
        executer(messages_en_attente())
      Fin Si
    Sinon
      attendre();
    Fin Si
  Fin Si
Fin Tant Que
```

Figure 4. Algorithme de fonctionnement des agents

On le voit dans cet algorithme, chaque agent se doit d'exécuter la fonction *executer(Messages_entrants)* dans laquelle doit être définie le rôle de chaque agent. Supposons qu'il faille utiliser un agent qui filtre un document en ne conservant qu'un document sur deux aléatoirement, voici ce que serait sa fonction *executer* :

```
Executer(messages_entrants)
  Pour Chaque message_entrant de messages_entrants de l'entrée 0
    Si aleat() > 0.5 Alors
      emettre message_entrant sur sortie 0
```

```
    Sinon
      emettre message_entrant sur sortie 1
    Fin Si
  Fin Pour
```

Figure 5. Exemple de fonction 'executer' pour un agent de filtrage aléatoire

Dans cet exemple simple, l'agent possède une seule entrée (numérotée 0) et deux sorties (numérotées 0 et 1).

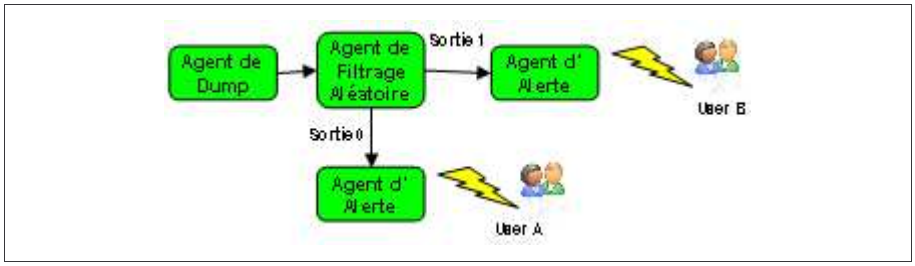

Figure 6. Exemple d'enchaînement pour un agent de filtrage aléatoire

La sortie 0 correspond aux documents filtrés positivement alors que la sortie 1 correspond aux documents filtrés négativement. Bien sûr, cet enchaînement est très simple et ne fait intervenir que très peu d'agents mais dans la théorie, tous types d'agents peuvent être enchaînés (récupération de lien, filtrage par mot-clé, agents de clustersation, etc…).

3.1.2. Les communications

Le schéma de communication de la plateforme est en fait très simple car il est essentiellement constitué des messages de type document. En fait, les agents vont travailler sur une instance de document et le transmettre de manière successive en ajoutant au fur et à mesure du traitement des connaissances sur les données issues des différents traitements.

3.2. Description des agents nécessaires

Le chapitre deux introduisait les étapes nécessaires à une veille. Les agents qui vont être décrits dans ce chapitre vont donc reprendre le schéma formel qui avait été donné. Seule la localisation et l'exploitation est laissée respectivement à la charge de l'administrateur et de l'utilisateur. On arrive donc à 4 types d'agents différents.

3.2.1. Les agents de dump

Ce sont eux qui vont aller chercher l'information sur les medias disponibles. Ils seront capables de télécharger des pages HTML, des mails, des fichiers et autres sources d'information utiles pour les veilleurs. A chaque information récupérée sera associée un

document *(Figure 1 en page 3*) unique[2] et qui pourra bien sûr être manipulé et échangé par les autres agents. En général ces agents possèderont une sortie et aucune entrée.

3.2.2. Les agents de stockage

Ils servent à sauvegarder un document localement et à identifier si le contenu de celui-ci a changé depuis la dernière sauvegarde. Pour cela, ils possèdent une entrée et une sortie.

3.2.3. Les agents d'alerte

Lorsqu'ils reçoivent un document, ils peuvent prévenir l'utilisateur final en envoyant par exemple un mail ou un SMS (c'est ce qui est appelé une alerte). Ils ne possèdent en général pas de sortie.

3.2.4. Les agents d'analyse

Ce sont les agents les plus importants du système puisque ce sont eux qui vont analyser les documents et pouvoir juger de la pertinence de l'information contenue dans ceux-ci. Leur nombre d'entrées et de sorties peut varier en fonction de l'outil (voire description des agents d'analyse dans le chapitre suivant).

3.3. Les agents d'analyse

Traditionnellement, les analyses textuelles se composent de deux parties distinctes qui sont les statistiques et la linguistique. La plateforme intègre ces concepts et une description de chacun est donnée ci-dessous.

3.3.1. Les agents statistiques

Ils se basent sur les statistiques des mots présents dans le texte et en particulier leur fréquence. Ainsi, il existe un certain nombre d'algorithmes pour traiter les documents et quelques détails sont donnés ci-dessous :

3.3.1.1. Prétraitements

Une manière simple et commune de représenter un document pour un ordinateur est d'utiliser une représentation vectorielle (Vector Space Model [Un Yong Nahm et Raymond J Mooney, 2001]). Ainsi, chaque document va être représenté par un vecteur dans l'espace vectoriel de tous les mots existants. Cependant, pour limiter la taille de cet espace, quelques traitements sont effectués :

– Seuls les mots appelés 'significatifs' vont être conservés (les mots courants comme les articles, les adjectifs, les sujets vont être supprimés). C'est ce qui est appelé le traitement par 'stop-list'.

2. Il possède une clé de hashage pseudo unique (MD5) basé sur l'Url et le contenu du document.

– Pour éviter de voir apparaître des mots qui ont la même racine (‘exploiter’ et ‘exploitation’ ont la même racine ‘exploit’ en commun), un algorithme de lemmatisation est appliqué au texte [Porter M.F.].

– Eventuellement, l’utilisation d’un dictionnaire peut éliminer certains mots qui ont la même signification.

– Utilisation des algorithmes de suppression des bruits.

A partir de cet instant, tout type de traitement statistique peut être appliqué.

3.3.1.2. Clusterisation

La clusterisation consiste à regrouper des textes en fonction du thème abordé à l’intérieur de celui-ci. C’est un sujet qui passionne les chercheurs depuis de nombreuses années et un certain nombre d’algorithmes ont été développés pour automatiser cette tâche [Yang Y et Liu X, 1999]. En voici quelques-uns :

– TFIDF Classifier, kNN (k Nearest Neighbor) : basé sur un calcul de distance entre documents.

– Bayes : basé sur une approche statistique (Cf. règles de Bayes).

– SVM (Support Vector Machine) : approche récemment reprise par Joachims.

Toutes ces méthodes offrent des résultats différents mais SVM semble se détacher du lot en offrant des performances supérieures [Joachims T., 1998].

3.3.2. Les agents linguistiques

La linguistique est une approche fréquemment utilisée dans l’extraction de l’information, en se basant sur des règles grammaticales pour analyser la structure des phrases en exploitant les liens et les dépendances entre les mots.

L’agent linguistique intégré dans cette plate forme de veille permet d’extraire des informations pertinentes autour de concepts choisis par l’utilisateur et les insérer dans une base de données. Nous utilisons un outil d’analyse syntaxique élaboré à l’université Henri Poincaré Nancy 1 dans le cadre de la thèse de Patrice Lopez (1999). Cet outil utilise le formalisme des grammaires lexicalisées d’arbres adjoints (LTAG) qui est un formalisme syntaxique dont l’élément de base est un arbre élémentaire. L’analyse d’une phrase à l’aide de ce formalisme donne comme résultat des arbres dérivés, qui fournissent les structures syntagmatiques associées à la phrase obtenue par combinaison des arbres élémentaires. Pour extraire les informations pertinentes nous prenons en considération tous les liens de substitution.

Les différentes étapes de l’analyse linguistique sont donc :

- Extraction des phrases pertinentes
- Analyse syntaxique des phrases retenues
- Extraction de la structure optimale à partir des structures syntagmatiques
- Insertion dans une base de donnée

Cette base de donnée ou table permet de fournir tous les évènements liés à la requête de l’utilisateur.

4. Extension du modèle multi agent

La description succincte qui a été faite de l'outil permet de se donner une idée du fonctionnement global de la veille automatisée. Cependant, il reste un certain nombre de points à améliorer et ceux-ci concernent tout particulièrement le retour d'expérience utilisateur et l'ergonomie (tous les agents sont au même niveau et présentent donc tous les détails de configuration au veilleur, ce qui peut être gênant si les options sont nombreuses).

4.1. La gestion du retour d'expérience utilisateur

Comme indiqué dans les chapitres précédents, l'utilisateur final doit pouvoir influencer les résultats de la veille sans pour autant intervenir directement sur le logiciel (ceci est à la charge de l'administrateur). Il existe deux types d'erreurs que peut commettre l'outil : les erreurs de dump et les erreurs d'analyse. Chaque cas est différent et une solution est proposée pour chaque.

4.1.1. Erreurs de dump

Ces erreurs sont liées à la présence de documents qui n'intéressent absolument pas l'utilisateur final, typiquement lorsque l'administrateur a fourni un pointeur vers un site qui ne concerne pas l'utilisateur (il a reçut une alerte à propos d'un document qui ne le concerne pas par exemple). Dans ce cas, la solution est d'éliminer le document dès qu'il est téléchargé (il est impossible de ne pas le télécharger car il faut connaître son contenu pour pouvoir l'analyser…). Pour cela, il faut intégrer un nouveau type d'agent nommé agent de contrôle : cet agent est chargé de récupérer le message de l'utilisateur (un mail dans lequel est indiqué le type de l'erreur et le document incriminé identifié par sa clé unique) et de localiser la source de l'erreur (le chemin parcourut par le document est connu). Sa tâche consiste ensuite à insérer entre l'agent incriminé et le suivant un agent de filtrage de document qui va bloquer les documents qui ressemblent trop à celui qui a été renvoyé par l'utilisateur (typiquement un filtre basé sur une méthode de Rocchio [Yang Y et Ault T; 2001]). Cet agent comportera une entrée et deux sorties (une pour le filtrage positif et un pour le négatif). Un l'algorithme pour les erreurs de ce type est fournit ci-dessous.

```
Tant Que vivant()
  récupérer_les_messages_en_attente();     // (les mails utilisateur)
  Pour Chaque message
    recupérer_le_document_et_sa_chaine_de_dump ();
    identifier_agent_qui_a_dumpé_document();
    insérer_entre_cet_agent_et_ses_successeur_agent_de_filtre_rocchio();       //
sauf s'il existe déjà
    insérer_dans_ce_filtre_le_document_incriminé();
  Fin Pour
Fin Tant Que
```

Figure 7. Algorithme de l'agent de contrôle pour les erreurs de type 'Dump'

4.1.2. Erreurs d'analyse

Ces erreurs concernent principalement le fait qu'un document est arrivé dans un agent alors qu'il aurait dû arriver dans un autre (cas de la clusterisation par exemple). Cela implique plusieurs obligations :

– Disposer d'une version agentifiée de l'algorithme de clusterisation.

– Pouvoir détourner un document à la volée pour le renvoyer vers un autre agent.

Le premier point est en cours de réalisation (donc non abordé dans ce papier) mais le second point dispose déjà d'une solution théorique qui est présentée ci-dessous :

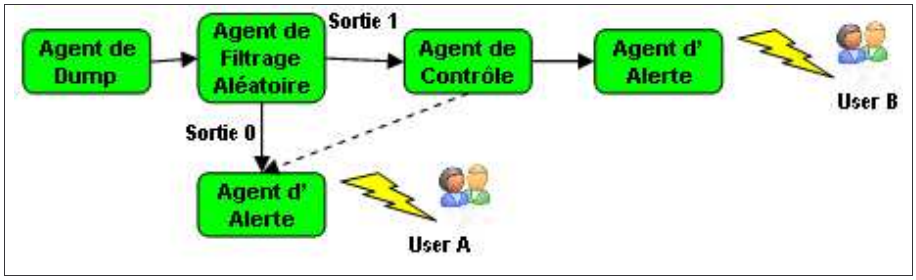

Figure 8. Gestion des erreurs de type 'analyse'

Dans le schéma ci-dessus, on peut voir qu'un agent de contrôle a été inséré en l'agent d'alerte de l'utilisateur B et celui de l'utilisateur A. En fait, cet agent va renvoyer une partie des documents destinés à B vers A. Ceci est possible de la même façon que pour les erreurs de dump et donc l'algorithme pour ce type d'erreur est presque le même :

```
Tant Que vivant()
  récupérer_les_messages_en_attente();     // (les mails utilisateur)
  Pour Chaque message
    recupérer_le_document_et_sa_chaine_de_dump ();
    identifier_agent_qui_a_mal_analysé_le_document();
    insérer_après_cet_agent_et_ses_successeurs_agent_de_filtre_rocchio();      //
sauf s'il existe déjà
    insérer_dans_ce_filtre_le_document_incriminé();
  Fin Pour
Fin Tant Que
```

Figure 9. Algorithme de l'agent de contrôle pour les erreurs de type 'Analyse'

On le voit au final, les deux types d'erreurs peuvent être gérées par le même type d'agent (à peu de changement près).

4.2. Macro agents

Les macro agents sont une réponse au problème d'ergonomie car ils permettent de présenter à différents niveaux de détails un système complexe.

4.2.1. Définition

Les macro agents sont des agents qui contiennent eux-mêmes des agents en interne. Leurs communications sont dirigées par les agents internes qui peuvent émettre des messages de l'intérieur du macro agent vers l'extérieur (voir *Figure 10 ci-dessous).*

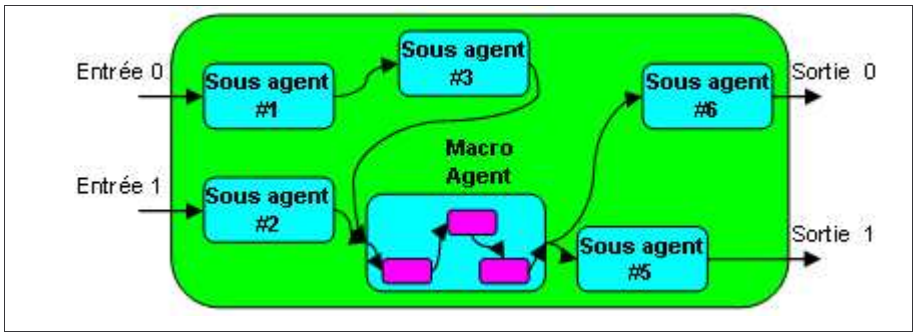

Figure 10. Exemple de macro agent (contenant un autre macro agent)

L'intérêt de cette représentation provient des options qui peuvent être présentées à l'utilisateur à différents niveaux. De la même manière que certains objets Java présentes des propriétés accessibles publiquement (Java Beans), les sous-agents vont pouvoir présenter des propriétés qui seront visibles ou non au niveau supérieur (cela permet de cacher à un niveau 'x' des données mais de pouvoir quand même y accéder en 'zoomant' au niveau 'x+1').

4.3. Séparation client / serveur

Dans un environnement multi utilisateur, il est souvent nécessaire de séparer le serveur et le client (pour éviter certains problèmes de sécurité et permettre un traitement en mode déconnecté). La solution retenue pour réaliser cette tâche repose sur l'utilisation des RMI Java. En effet, d'un côté les agents vont fonctionner de manière indépendante dans un environnement multi agent (i.e. le serveur) et d'un autre côté, le client va venir se connecter à ses agents pour pouvoir les configurer. Une mise à jour de l'interface graphique va permettre de configurer l'agent, et un changement d'état de l'agent va influencer l'apparence de l'interface graphique : c'est le modèle client / serveur bidirectionnel.

Pour cela, les agent ET les interfaces devront disposer d'une interface RMI standardisée qui leur permettra de communiquer directement entre eux (cette méthode est beaucoup plus efficace et pratique que l'utilisation de messages KQML ou autre).

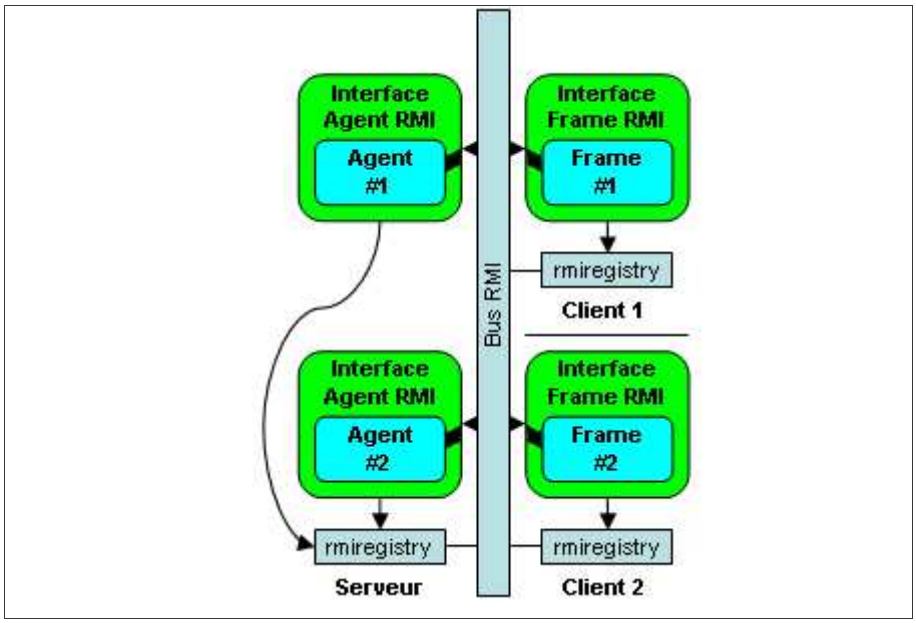

Figure 11. Utilisation de RMI pour la distribution client / serveur

5 Conclusion et évolutions

Ce papier a présenté une nouvelle architecture basée sur un système multi agent guidé par l'utilisateur. Elle introduit aussi un certain nombre de concepts pour gérer le retour d'expérience utilisateur ainsi que la distribution des interfaces graphiques. Les tests effectués par des veilleurs professionnels montrent que la plateforme définie répond bien aux besoins d'adaptabilité mais souffre d'un certain nombres de problèmes qui sont de différentes natures :

Potentiel des agents inaccessibles aux novices (les agents sont parfois difficiles à configurer)

Notion d'agents et de flux pas toujours maîtrisé par l'utilisateur.

Les évolutions naturelles à envisager sont donc encore d'améliorer l'ergonomie pour simplifier l'usage et la configuration de l'outil (Ce n'est plus un problème de présentation des données mais bel et bien de simplification/automatisation de la configuration).

Références

[Goujon B, 2000] Goujon B, Utilisation de l'exploration contextuelle pour l'aide à la veille technologique, Thèse de doctorat, Université de Paris IV Sorbonne, 2000.

[Denis X, 2001] Denis X, Une Architecture Multiagent pour la Veille Informatique sur Internet, Rapport interne, Laboratoire d'Informatique du Havre, 2001.

[VEI02] http://www.veille.com

[Goebel et Le Gruenwald,1999] Goebel M., Le Gruenwald, *A Survey of data mining and Knowledge Discovery Software Tools*, University of Auckland, 1999.

[Ferber J, 1995] Ferber J., Les systèmes multi-agents. Vers une intelligence collective, Inter-Éditions, 1995.

[Un Yong Nahm et Raymond J Mooney, 2001] Un Yong Nahm and Raymond J. Mooney, Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI01), pages 979984, Seattle, WA, August, 2001

[Porter M.F.] Porter M.F., Snowball: A language for stemming algorithms, http://snowball.tartarus.org/texts/introduction.html.

[Yang Y et Liu X, 1999] Yang Y. and Liu X., A re-examination of text categorization methods. Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), 1999, pp 42--49. Carnegie Mellon University.

[Joachims T., 1998] T. Joachims, Text Categorization with Support Vector Machines: Learning with Many Relevant Features, Dortmund University, 1998.

[Yang Y, 1995)] Y. Yang, Mayo, Noise Reduction in a Statistical Approach to Text Categorization, Clinic (Minnesota), 1995.

[Yang Y et Ault T; 2001] Yang Y. and Ault T., kNN, Rocchio and Metrics for Information Filtering at TREC-10. Carnegie Mellon University. TREC-10 Notes, Nov. 2001.

Summary

The Internet growth has favoured the apparition of online data, giving an important opportunity for business intelligence tools Unfortunately, there is so much information available that's its impossible for a human being to keep in touch with everything. Some tools are available to help users in such a task but few of them are really adaptive. The software that is presented here relies on a multiple agent system where each entity plays a part in the business intelligence process (dumping, analysis, learning), the innovating part coming from the management of the agent communications by the user.