Utilisation collaborative d'outils de textmining pour la veille sur Internet

Xavier Denis [*,], Gaele Simon [*], Stephan Brunessaux [**]**

xavier.denis@operamail.com, gaele.simon@iut.univ-lehavre.fr, stephan.brunessaux@sysde.eads.net

[*] Laboratoire d'Informatique du Havre
UFR Sciences & Techniques
25 Rue Philippe Lebon, 76600 Le Havre
http://www-lih.univ-lehavre.fr

[**] EADS S&DE
Parc d'Affaires des Portes
BP613, 27106 Val-de-Reuil Cedex
http://www.eads.com

Mots-clés :
Text mining, système multi-agents, MadKit, collaboration, peer-to-peer, modularité, générique.

Keywords :

Text mining, multiple agents system, MadKit, Collaborative work, peer-to-peer, modularity, generic.

Palabras clave :

Text mining, sistema autoadaptativa, MadKit, trabajo colaborativo, P2P, genérico.

Résumé :

L'essor d'Internet ces 10 dernières années a permis de mettre à la disposition de tous une grande quantité d'information accessible en ligne. Or certaines entités comme les PME et PMI sont très intéressées par ces nouvelles sources de données qui leur permettraient de mieux surveiller l'activité de leurs concurrents (arrivée de nouveaux produits, dépôts de brevets, ...). Ainsi est née la problématique de la veille qui est de scruter une source de données afin de détecter un ensemble d'information utiles. Bien sûr il existe un certain nombre d'outils disponibles sur le marché (principalement commerciaux) mais leur conception fermée ne permet pas de développer facilement de nouveaux modules dédiés à la surveillance d'un site ou d'une thématique non prévue à l'avance. Il existe donc un besoin concernant la mise à disponibilité d'un outil de veille qui soit ouvert et

configurable par un utilisateur et qui permette d'intégrer facilement de nouvelles technologies sans remettre en cause celles déjà employées. Notre approche, basée sur un système multi-agents, propose à l'utilisateur une plate-forme client/serveur mettant à sa disposition un certain nombre de modules de base (récupération des données, filtrage, alerte) que l'utilisateur devra agencer afin de répondre à sa problématique de veille. Chaque module est représenté par un agent autonome et c'est à l'utilisateur de définir le réseau d'accointances de ces agents afin de créer le flux de traitement nécessaire à la bonne réalisation du travail. L'intérêt des agents est ici de pouvoir gérer le retour d'expérience de l'utilisateur en fonction de ses actions entreprises lors de la création du flux et de la consultation des résultats de la chaîne de traitement des agents. Plus précisément, il s'agit de mettre à la disposition des utilisateurs un outil qui leurs permette de répondre aux problématiques de veille et dans lequel des agents travaillent de manière coopérative sur des documents issus d'Internet.

1 Introduction

L'essor d'Internet ces 10 dernières années a provoqué un besoin grandissant dans la surveillance et l'analyse de son contenu informatif. De nombreux logiciels ont donc vus le jour et qui permettent de réaliser certaines opérations de veille stratégique telles qu'elles sont généralement décrites dans la littérature [3] : la récupération, l'analyse et la distribution de l'information définissent en effet le principe du cycle de la veille [1].

Parallèlement à cela, de nombreuses méthodes majoritairement développées par des universitaires sont apparues afin de répondre à certain besoins éparses dans le traitement de l'information comme par exemple la classification (SVM et autres méthodes récentes [11] ou bien l'analyse linguistique [9]).

Malheureusement, ces méthodes sont souvent méconnues et relativement peu accessibles. De plus, la représentation de l'information manipulée n'étant pas uniforme (souvent limitée à une chaîne de caractère ou un texte brut), l'utilisation conjointe de ces outils est impossible sans une ré-écriture complète de l'algorithme.

Notre idée est donc de proposer une plate-forme de veille collaborative dans laquelle des modules, mis à disposition de tous, sont utilisés pour réaliser un flux de traitement distribué. L'approche innovante consiste à utiliser des agents autonomes [6] pour réaliser ces modules et qui sont capables de gérer le retour d'expérience de l'utilisateur lorsque cela est nécessaire. Afin de répondre au besoin d'uniformisation de l'information, les documents manipulés sont transformés au format XML TEI et pour lesquels il est possible d'y adjoindre des annotations créés par les agents lors de leur parcours le long de la chaîne de traitements.

2 Architecture générale

Avant de décrire comment fonctionne la plate-forme et comment sont représentés les documents manipulés, il faut tout d'abord revenir au contexte de la veille, et comment celle-ci est réalisée.

2.1 Contexte de la veille

Afin de simplifier le problème, supposons que la veille soit réalisée par une seule personne et que celle-ci appartienne à une communauté de veilleurs. Nous obtenons donc n veilleurs {V1, …, Vn} qui réalisent des tâches de veille et qui peuvent se partager leurs expériences et leurs méthodes.

Leur objectif est donc de créer des chaînes de traîtements qui répondent à leurs besoins de veille et de les partager afin que d'autres veilleurs qui auraient une problématique similaire puisse les ré-utiliser simplement. Le réseau informatique (Internet) étant le meilleur moyen de communication, nous obtenons le schéma de fonctionnement de la Figure 1.

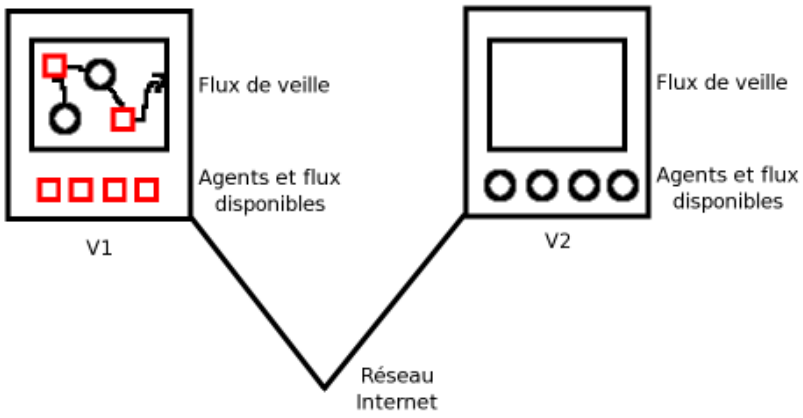


Figure 1 : Contexte distribué de la veille (exemple avec 2 veilleurs)

Dans cet exemple, V1 possède des agents spécialisés dans le téléchargement de ressources (carrés rouges) et V2 possède des agents capables de filtrer les documents en fonction de leur contenu (ronds noirs). Alors V1 va vouloir utiliser les agents proposés par V2 afin de réaliser sa tâche de veille : télécharger des pages et les filtrer en fonction de leur contenu. Pour cela, V1 devra créer un flux d'agents (dont la description formelle est donnée dans le paragraphe suivant) utilisant des agents disponibles en local et des agents disponibles de manière distribuée sur la plate-forme de V2.

Ce schéma est finalement très proche du fonctionnement des réseaux peer-to-peer [8, 10]. En effet, chaque veilleur va se connecter aux autres afin de détecter les ressources disponibles (ici des agents) et les utiliser à la volée dans leurs propres chaînes de traitement. L'idée est de ne pas passer par un serveur centralisé qui contiendrait toutes les agents disponibles mais d'utiliser les capacités de distribution afin de réaliser des connexions client/client directes et surtout de permettre à des veilleurs de partager leurs agents à la volée.

Contrairement aux approches classiques de veilles qui sont orientées multi-clients/mono-serveur, notre approche est beaucoup plus dynamique et autonome puisqu'elle se base sur une approche multi-clients/multi-serveurs dans le cadre du partage des ressources.

2.2 Flux d'agents

Le flux d'agents est défini ici comme une chaîne de traitements dans laquelle des agents autonomes [6] sont instanciés par un utilisateur et reliés de façon à former un flux où circulent des documents qui seront traités. Cette approche qui peut paraître inhabituelle consiste à fixer une partie du réseau accointances des agents afin de créer un chaînage logique de traitement.

Plus formellement, il s'agit de spécialiser une partie des moyens de communication des agents en ajoutant la gestion des communications par flux statique (Figure 2).

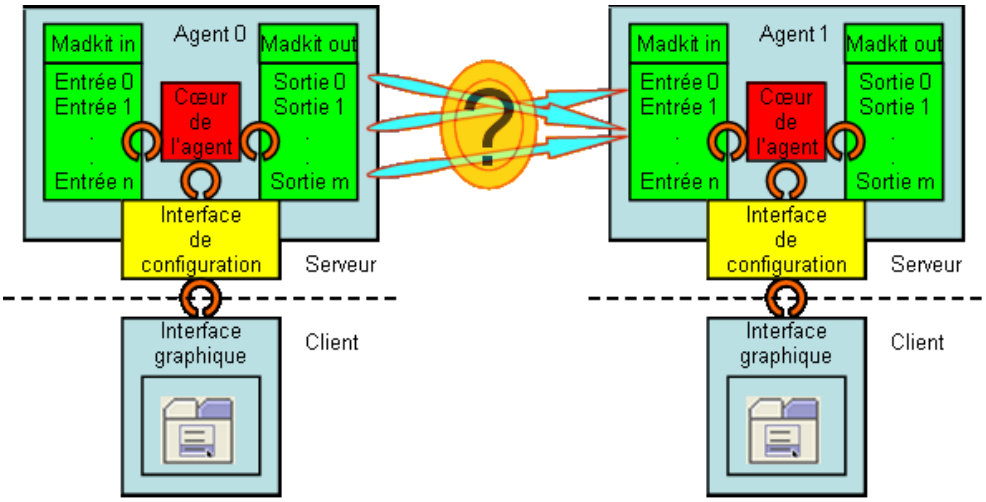


Figure 2 : Structure des agents du flux

L'agent possède donc 2 types de canaux de communications :

1. Les canaux dédiées au flux numérotés de 0 à n. Il peut en exister plusieurs afin de créer des chaînages ayant une symbolique différente : un agent de filtrage par mots-clés va posséder 2 sorties, une pour les documents positifs, et une pour les documents négatifs.
2. Les canaux classiques (gérées ici par MadKit [4]) utilisés pour les communications génériques avec les autres agents (hors flux).

Le but d'un agent est donc, entre autres, de réagir à des évènements extérieurs qui peuvent être multiples :

- Arrivée d'un nouveau document sur l'un de ses canaux.
- Interaction de l'utilisateur avec l'interface graphique.
- Arrivée de nouveaux messages sur les canaux d'entrées classiques.

Bien que très réactive en apparence, cette façon de gérer le flux est guidée par une approche pro-active des agents. En effet, il s'agit de pouvoir gérer le retour d'expérience utilisateur lors de l'arrivée de documents (d'où l'intérêt d'utiliser des agents).

Prenons le cas d'un agent qui va classifier les documents (i.e. qui va posséder n sorties, chacune d'entre elle étant associée à un cluster). Les méthodes de classification automatiques étant basées sur des heuristiques [5], l'utilisateur doit pouvoir préciser les erreurs commises afin que le système

puisse les prendre en compte. Typiquement, cela se fera par apprentissage de l'agent lors de l'interaction (au moyen de l'interface graphique) avec l'utilisateur : celui-ci déplacera certains documents d'un cluster à un autre afin de 'faire comprendre' à l'agent les erreurs commises. Lors de l'arrivée d'un nouveau document sur cet agent, les actions précédentes prises par l'utilisateur permettront à l'agent de mieux le catégoriser. Cette méthode a été précédemment exposée à EGC'04 [2].

2.3 Besoins de la veille

Les besoins de la veille vont influer directement sur les capacités des agents disponibles et sur leurs fonctionnalités. Typiquement, il y a 3 grandes étapes dans le déroulement de la veille [1] :

1. La récupération des informations : étape initiale dans laquelle des données (pages Web, documents Word, ...) sont ramenées en local à des fins d'archivage et de comparaison avec d'autres révisions (une révision étant une ressource identifiée par une URL et dont le contenu change au cours du temps).
2. L'analyse des informations : étape transitoire dans laquelle les documents sont traités pas des agents afin de réaliser un filtrage/catégorisation/résumé voulu par l'utilisateur.
3. L'envoi des informations : étape finale dans laquelle les documents sont envoyés au veilleur intéressé.

Ces 3 grandes étapes définiront donc les 3 types d'agents disponibles dans la plate-forme pour réaliser le flux de traitement :

1. Les agents 'de dump' : agents placés en début de flux chargés de récupérer des sources d'informations.
2. Les agents 'd'analyse' : agents placés au centre du flux chargés de filtrer/annoter/rediriger les documents. Dans le cadre de la réalisation de projets, des agents ont été développés afin de répondre à certaines problématiques : filtrage par mots-clés, filtrage par centre d'intérêt, récupération automatique de liens, identification de notions telles que les noms propres, adresses, classification automatique, clustering, ...
3. Les agents 'd'alerte' : agents placés en fin de flux et chargés d'envoyer les documents au veilleur.

Le veilleur aura donc le choix de créer des chaînes de traitement à partir de ces agents disponibles et dont certains peuvent appartenir à une machine distante (donc mis à disposition par d'autres veilleurs).

2.4 Organisations d'agents

Afin de répondre à l'architecture du peer-to-peer, les agents doivent être organisés de manière un peu spéciale avec, en particulier, des agents gérant les communications entre les différentes plates-formes (clients et serveurs) et possédant des groupes et rôles spécifiques. Ainsi, la plate-forme dispose de 4 types d'agents :

- Les agents du flux tels qu'ils ont été décrits dans le paragraphe précédent.
- Les agents de communications du côté serveurs : agents spécialisés dans la gestion de la connexion et des communications entre les serveurs et les clients.
- Les agents de communications du côté clients : agents spécialisés dans la gestion de la connexion et des communications avec les serveurs.
- Des agents appelés macro-agents dans lesquels un veilleur peut instancier les agents du flux (ce sont donc des conteneurs d'agents). Leur particularité réside dans le fait que les messages broadcast émis par des agents appartenant à ce macro-agent ne peuvent pas être reçus par des agents à l'extérieur de celui-ci. Cela permet de créer des environnements privatifs et autonomes afin de définir des flux qui n'interfèrent pas entre eux.

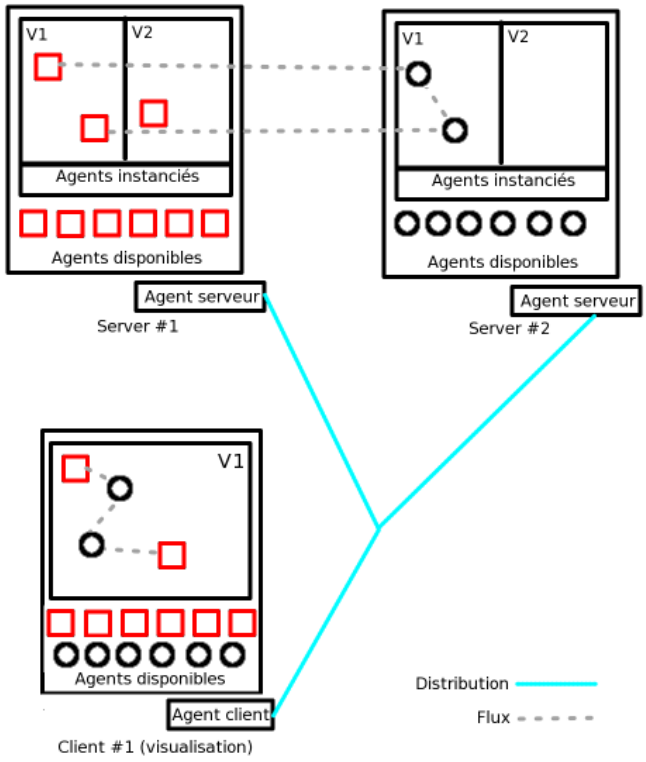


Figure 3 : Organisation et connexion entre agents

La Figure 3 décrit la façon dont la distribution s'opère.

Les veilleurs disposent sur chaque serveur d'un espace de travail dans lequel chacun peut instancier les agents disponibles. Il faut remarquer que l'instanciation d'un agent ne peut se faire que sur le serveur où il est proposé puisqu'il peut dépendre de ressource locales auxquelles le veilleur n'a pas accès (Thésaurus ou dictionnaires par exemple). Ainsi, le client se connecte aux serveurs préalablement spécifiés par le veilleur et reconstruit graphiquement la chaîne qui est distribuée sur plusieurs machines.

On le voit bien dans la figure, les agents de communication ont un rôle pivot dans la distribution. En effet, ce sont les agents de communication du côté serveur qui sont distribués et possèdent un groupe et un rôle fixe (dans notre cas, ils appartiennent tous au groupe 'server' et possèdent tous le rôle '*'). Ainsi, lorsqu'un client se connecte, il interroge tous les agents pour connaître ceux qui appartiennent à 'server/*'. La liste retournée correspond donc à la liste des serveurs disponibles sur le réseau et dans lesquels le veilleur va pouvoir instancier des agents. L'agent de communication client va donc avoir la tâche de récolter les informations des serveurs (noms, agents disponibles, flux créés, ...) afin de reconstruire le flux et permettre au veilleur d'instancier de nouveaux agents comme ci ceux-ci étaient disponibles en local.

Le client n'a donc qu'un rôle de visualisation puisque les traitements effectifs par les agents sont réalisés sur les serveurs.

Plus formellement, du côté du serveur, les agents sont organisés de manière pyramidale (Figure 4) :

- Au sommet se trouve l'agent de communication serveur (qui est dans le groupe distribué 'server' et qui possède le rôle '*').
- En dessous se trouvent les macros-agents qui définissent les environnements disponibles pour les veilleurs. Ils appartiennent au groupe 'RootMacroAgents' et leur rôle est le nom (login) du veilleur. Ce groupe n'est pas distribué car si le client veut connaître les macros-agents instanciés par le veilleur V2, il devra forcément interroger l'agent serveur (qui peut vérifier par exemple si le login et le mot de passe fourni par le client sont corrects).
- Encore en dessous se trouvent les agents instanciés par l'utilisateur pour réaliser le flux. Ils sont dans le groupe dont le nom est celui du macro-agent qui les contient. Leur rôle est '*'.

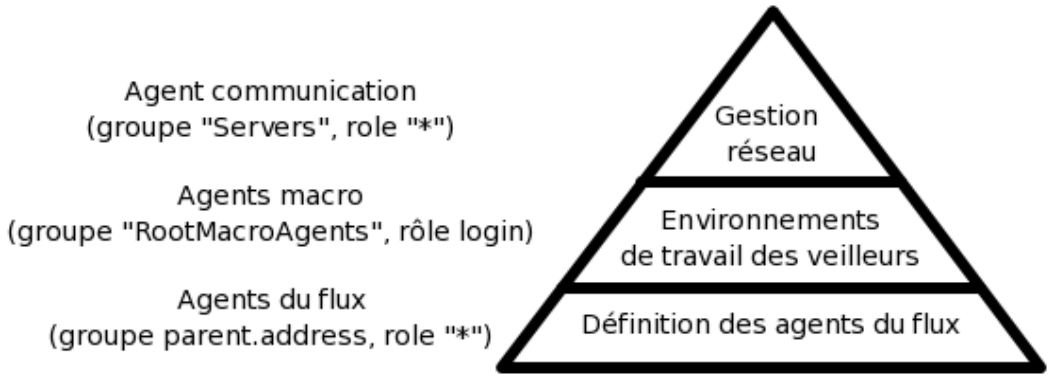


Figure 4 : Organisation des agents en pyramide

Cette architecture en pyramide permet une meilleure sécurité car le passage obligé pour la connexion est l'agent de communication. Dans l'implémentation qui a été faite de la plate-forme, tous les messages qui circulent possèdent un login et un mot de passe que cet agent est chargé de vérifier afin d'authentifier un veilleur qui utilise la plate-forme.

2.5 Description des documents

Les documents sont les ressources que les agents vont s'échanger sur le flux statique défini par l'utilisateur.

Dans le cadre de la réalisation d'une plate-forme expérimentale pour le projet du ministère de l'industrie nommée XMIner, la norme TEI [7] a été retenue afin de décomposer le document en unités de traitements simples (propriétés [titre, auteur, ...], paragraphes, phrases).

Un autre format propriétaire (appelé 'pivot') a été utilisé afin de rajouter la possibilité d'annoter le document (c'est à dire marquer certaines portions du texte comme contenant des informations) ; il est basé sur l'utilisation d'XML et la décomposition du contenu textuel du document en mots repérés par un identifiant unique (voir exemple dans la Figure 5).

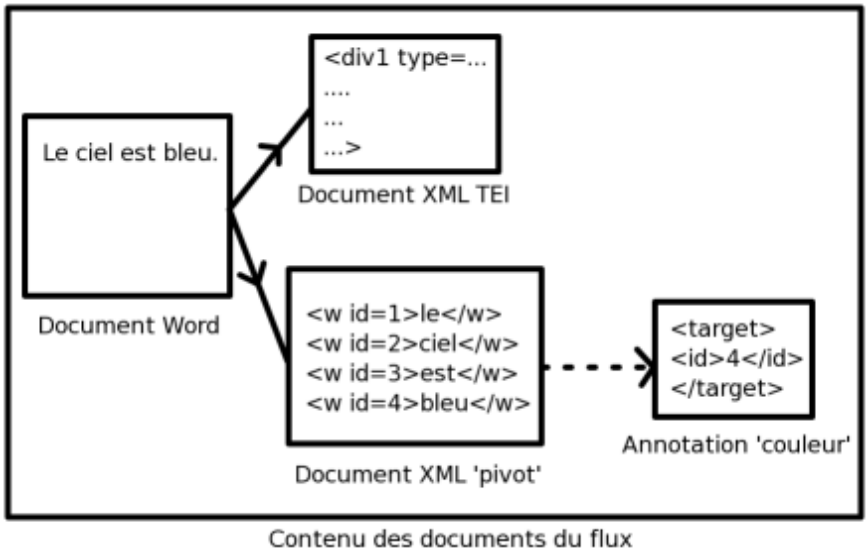
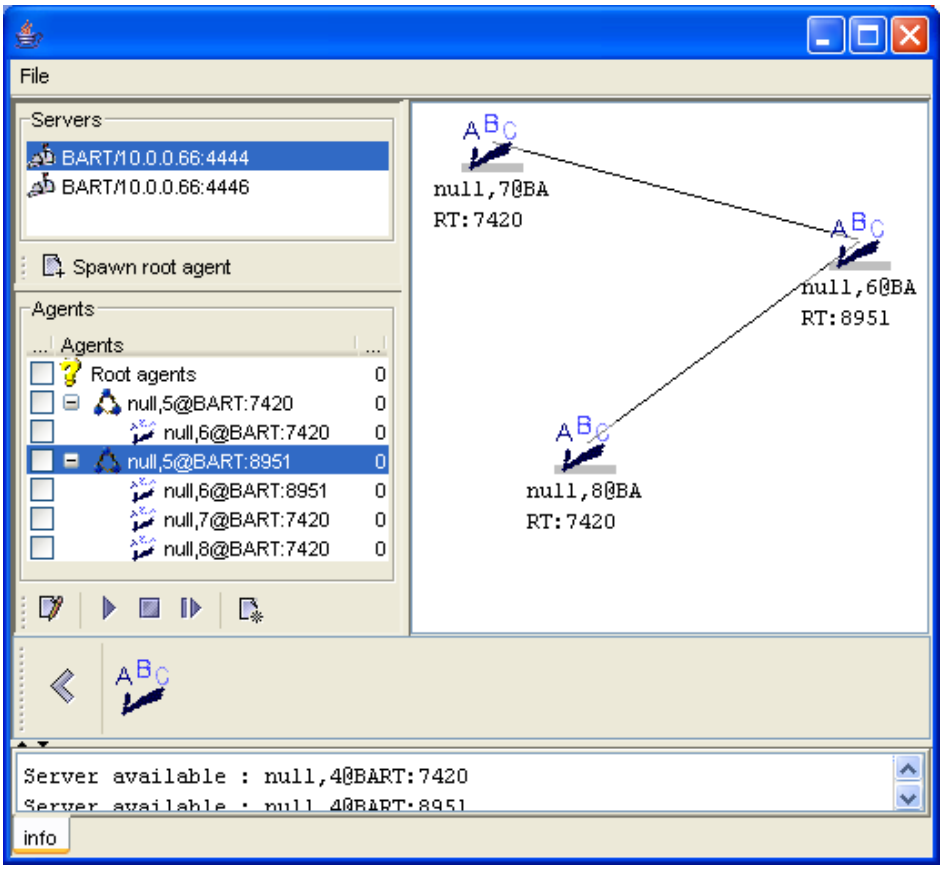


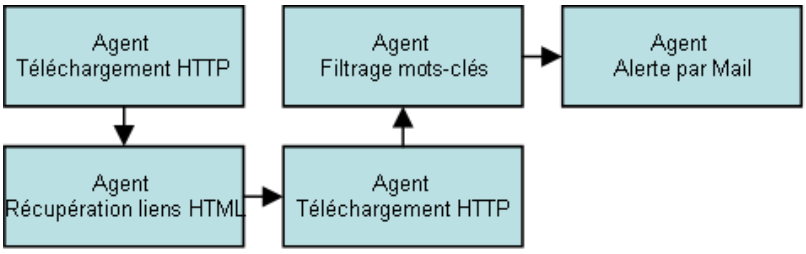
Figure 5 : Exemple d'annotations d'un document

2.6 Capture d'écran et exemples de flux

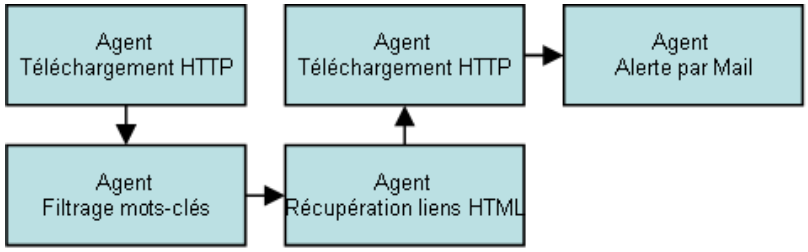
Une capture d'écran ci dessous présente la plate-forme (côté client) telle qu'elle existe actuellement.



Ci dessous est présenté un flux capable de filtrer les liens n+1 dans une page en fonction de mots-clés contenus dans ces pages.



Ci dessous est présenté un flux capable de filtrer les liens n+1 dans une page en fonction de mots-clés contenus dans les pages de niveau n+1.



Les exemples ci-dessus montrent que la réorganisation du flux permet d'obtenir des résultats différents. Le principe de flux permet donc d'avoir un outil modulaire et évolutif ce qui fait que cette plate-forme se situe plutôt dans la catégorie des méta-outils de veille.

3 Evaluation et tests

L'évaluation de cette plate-forme s'est faite en deux parties :

1. Evaluation globale (subjective) de son utilisation par des veilleurs professionnels (dans notre cas des veilleurs appartenant à la CCI de l'Eure). Elle consiste en particulier à présenter les points forts et les points faibles du flux de traitement.
2. Evaluation qualitative des outils mis à disposition et en particulier de la gestion du retour d'expérience pour l'agent de classification.

3.1 Evaluation globale

L'évaluation globale consiste à donner l'outil à un utilisateur et à noter ses appréciations. Dans le cadre du projet PMIner, l'évaluation a été faite par des veilleurs de la CCI de l'Eure. Elle consistait à définir les capacités de l'outil grâce à l'utilisation du flux d'agents et les limites qui pouvaient exister.

De cette étude sont ressortis plusieurs points positifs et négatifs :

3.1.1 Points positifs

- Le flux permet une très grande modularité.
- L'approche par agents permet une meilleure visualisation des traitements en cours.

3.1.2 Points négatifs

- Flux parfois compliqué à mettre en place.
- Besoin de configurer les agents un par un avant d'obtenir une chaîne valide de traitement.

3.2 Evaluation qualitative

Certains tests unitaires ont été effectues afin de vérifier les performances des agents pendant les phases d'analyse. Dans notre cas, c'est l'agent de classification (qui repose sur Single Pass et TFIDF) avec gestion du retour d'expérience qui a été évalué sur une partie du corpus Reuters.

Les tests [2] ont montré que l'algorithme obtenait de bons résultats (même par rapport à SVM) et qu'une gestion simple du retour d'expérience utilisateur permettait encore d'améliorer la qualité de classification.

4 Conclusion

La plate-forme qui est présentée dans cet article est en cours de développement. Certaines de ses fonctionnalités ont été évaluées et testées en condition réelle (la validité et l'intérêt du flux avec le projet PMIner et la classification avec le projet XMiner).

La dernière innovation présentée concerne le déploiement peer-to-peer et même si des tests complets n'ont pas encore été réalisés, les fonctionnalités apportées par cette approche promettent une plus grande diffusion et utilisation des outils de veille et de textmining en général.

Il reste néanmoins à proposer un outils complet (et non basé sur une plate-forme expérimentale) qui puisse être complètement validé par une communauté d'utilisateurs professionnels et qui propose d'autres fonctionnalités comme la construction d'une base de données distribuée de documents et la possibilité d'y effectuer des recherches.

5 Bibliographie

[1] Michel Cartier. La veille intégrée. Définition de la veille en 3 étapes, 1996.

[2] Xavier Denis. Approche innovante pour la recherche et l'extraction coopérative d'informations sur Internet. Dans RNTI – EGC'04, 2004.

[3] Bénédicte Goujon. Utilisation de l'exploration contextuelle pour l'aide à la veille technologique. PhD thesis, Université Paris IV – Sorbonne, 2000.

[4] Olivier Gutknecht et Jacques Ferber. The MADKIT agent plateform architecture. Dans Agents Workshop on Infrastructure for Multi-Agent Systems, pages 48-55, 2000.

[5] Thorsten Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. Dans Dougnlas H. Fisher, editor, Proceedings of ICML-97, 14th International Conference on Machine Learning, pages 143-151, Nashville, US, 1997.

[6] P. Maes. Modeling adaptive autonomous agents. Artificial Life, I. 1994.

[7] D. McKelvie, C. Brew, and H. Thompson. Using sgml as a basis for data-intensive natural language processing, 1998.

[8] Wee Siong Ng. PeerDB : a p2p-based system for distributed data sharing, 2003.

[9] Bärbel Ripplinger and Paul Schmidt. Automatic multilingual indexing and natural language processing.

[10] Beverly Yang and Hector Garcia-Molina. Comparing hybrid peer-to-peer systems. In The VLDB Journal, pages 561-570, sep 2001.

[11] Yiming Yang and Xing Liu. A re-examination of text categorization methods. In Marti A. Hearst, Fredric Gey, and Richard Tong, Proceedings if SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval, pages 42-49, Berkeley, US, 1999.