Approche innovante pour la recherche et l'extraction coopérative et dynamique d'informations sur Internet

Xavier Denis*,**, Gaële Simon*
Nicolas Chanchevrier**

*UFR Sciences & Techniques, 25 Rue Philippe Lebon, 76600 Le Havre
xavier.denis@operamail.com, gaele.simon@iut.univ-lehavre.fr
http ://www-lih.univ-lehavre.fr
**EADS S&DE, Parc d'Affaires des Portes BP613, 27106 Val De Reuil Cedex
nicolas.chanchevrier@tiscali.fr
http ://www.eads.com

Résumé. Il existe de nombreuses techniques qui permettent de classifier des documents textuels en fonction du centre d'intérêt d'un utilisateur (kNN, SVM, ...). Malheureusement, l'intégration de ces méthodes dans des plate-formes de textmining est souvent très statique et ne permet pas facilement d'affiner les traitements et/ou résultats au cours du temps. Le but de cet article est de présenter une plate-forme de webmining dans laquelle les données hétérogènes sont représentées uniformément selon un formalisme XML/TEI et où l'utilisateur peut interagir sur les processus de récupération et d'analyse de ces données. Pour cela, les modules de traitements sont représentés par des agents fonctionnant sur la plate-forme MadKit et l'apprentissage se fait sur une méthode dérivée de VSM[1] et TFIDF utilisant un principe de listes noires pondérées permettant la reconnaissance de documents indésirables. La dynamique de la plate-forme repose principalement sur la possibilité d'ajouter à la volée des agents de traitement et de pouvoir modifier l'ordre et les paramètres d'analyse des documents.

1 Introduction

La richesse des informations disponibles en ligne [Woodruff *et al.*, 1996] et leur diversité de contenu a ouvert la porte à un besoin grandissant ces dernières années : la recherche, l'analyse et la distribution d'informations. Ces trois grandes étapes définissent ce qui est communément appelé un processus de veille [Goujon, 2000]. Des logiciels existent déjà pour répondre à un certain nombre de ces besoins mais leur conception généralement fermée (API obscure et non documentée, outils commerciaux, ...) limite leur utilisabilité.

L'approche qui a été retenue ici est basée sur un système multi-agents MadKit [Gutknecht et Ferber, 2000] où chaque agent, relié ou non à un autre (principe de flux d'agents décrit au paragraphe 2.4), participe à chaque étape du processus de veille. Bien qu'assez simple en apparence, ce principe de flux implique une grande souplesse

[1] Vector Space Model

d'utilisation d'autant que les documents qui peuvent être amenés à être analysés sont représentés dans un format XML et conformes à la norme universelle de représentation des documents appelée TEI (voir paragraphe 2.3).

2 Description de la plate-forme

La plate-forme qui est décrite dans cet article est en cours de développement au sein du laboratoire de développement d'EADS S&DE basé au Val-de-Reuil (Eure) et au Laboratoire d'Informatique du Havre (Seine-Maritime). Les paragraphes qui suivent vont donc décrire le fonctionnement de celle-ci et les choix technologiques qui ont été retenus.

2.1 Objectifs

Le but de la plate-forme est de créer des environements de veille dans lesquels des agents téléchargent, analysent et redistribuent des documents récupérés sur diverses sources (Internet, fichiers Word, ...). La problématique n'est pas liée à la localisation de ces sources mais plutôt à l'analyse des pointeurs qui vont permettre d'extraire des informations interessantes. Ainsi c'est à l'utilisateur (personne interessés par la surveillance d'une source) de définir ce qu'il recherche en utilisant un principe de flux dans lequel les documents téléchargés sont annotés (informations ajoutées) à la volée.

2.2 Architecture générale

La plate-forme présentée est basée sur un nouveau concept de communications d'agents guidées par l'utilisateur : en effet, le but est de créer un flux où les agents logiciels sont connectés statiquement par l'utilisateur (principe équivalent à Clementine ou TextVis [Landau *et al.*, 1998]). Les agents communiquent en envoyant des messages correspondant à des documents XML (paragraphe 2.3) et travaillent dessus en modifiant/ajoutant des marqueurs relatifs à des informations trouvées. Cette approche n'empêche pas les agents d'utiliser les schémas traditionnels de communication classique par messages broadcasts ou messages par rôle [Ferber et Gutknecht, 1998].

Dans l'implémentation qui a été faite de cette plate-forme, il existe 3 types d'agents qui permettent de réaliser une veille complète ou même des opérations plus simples comme le téléchargement d'un site Web complet :

- Agents de 'dump' : agents utilisés pour se connecter à une source de données et récupérer son contenu (Internet, e-mails, ...).
- Agents d'analyse : agents utilisés pour analyser le contenu des documents (Certains ajoutent des informations, d'autres filtrent, ...).
- Agents d'alerte : agents utilisés pour envoyer une alerte à un utilisateur (par mail, SMS, ...).

Le principe de flux dérive avant tout de la façon dont les agents vont pouvoir s'interconnecter entre eux. En effet, les agents disposent d'entées/sorties qui vont servir à créer ce flux et les possibilités de connexions de ces entrées/sorties vont définir les possibilités de connexion des agents. Les agents de 'dump' possèderont en général une

seule sortie, les agents d'analyse auront une entrée et au moins une sortie et les agents d'alerte une seule sortie. Cela permet donc d'éviter la création de chaînes invalides (à quoi servirait un agent d'alerte avant un agent de 'dump' ?).

Le but d'un agent est donc (entre autres) de récupérer un document (en entrée), d'y ajouter des informations (liées à son traitement), et ensuite de le redonner à l'agent suivant (en sortie). Les documents sont décrits dans le paragraphe 2.3.

2.3 Les documents XML

Comme indiqué précedemment, les agents s'échangent (sur les canaux de communication représentant le flux) des informations sous forme de documents XML. Ces derniers contiennent en fait des propriétés (exposées ci-dessous) utilisées pendant le parcours du document dans le flux d'agents :

- Contenu binaire : c'est le contenu du document original téléchargé.
- Contenu textuel : c'est le document transformé au format XML/TEI.
- Annotations : ce sont les informations rajoutées par les agents/utilisateurs et qui pointent vers des parties du document XML/TEI.
- Date, URI, ... : autres informations pouvant être utiles pour les traitements.

La représentation sous forme XML/TEI [Sperberg-McQueen et Burnard, 2002] permet de manipuler n'importe quel format de fichier textuel en utilisant des balises standardisées de représentation du document [McKelvie *et al.*, 1997]. Ainsi il est possible, à partir d'un document TEI valide, de le transformer en document Word, PDF ou même Postscript en utilisant un fichier de conversion CSS. Il est donc théoriquement possible d'analyser le contenu textuel d'un document de n'importe quel type.

L'autre intérêt d'utiliser un format de représentation XML est de pouvoir ajouter des annotations sur ce document [Brew, 2000]. En effet il est possible, à partir des balises du document original, de faire 'pointer' une annotation vers une partie de ce document (technologie XPath/XPointer définie par le W3C[2]). A charge aux agents d'analyse d'ajouter ces annotations lors de la circulation du document dans le flux.

2.4 Le flux d'agents

Dans les approches traditionnelles orientées multi-agents, le réseau d'accointances des agents se construit généralement de deux manières différentes :

1. L'agent est conçu pour répondre à des requêtes prédéfinies (correspondant à une approche réactive).
2. L'agent contruit son réseau d'accointances en fonction des résultats obtenus lors de ses collaborations avec d'autres agents (suivant son rôle/groupe comme le définit Ferber ce qui correspond à une aproche pro-active).

Dans ces deux approches, l'utilisateur est souvent passif et ne peut donc pas intervenir efficacement dans le fonctionnement de l'agent. Or dans la cas de la veille, il est primordial que l'utilisateur puisse modifier les comportement des agents pour obtenir les résultats qu'il souhaite. Pour répondre à cette exigence l'approche retenue

[2]World Wide Web Consortium.

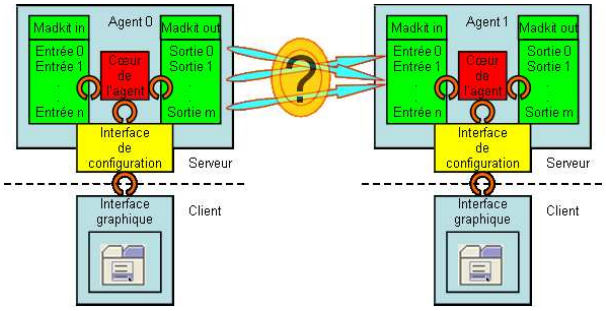

FIG. 1 – Architecture des agents et principe de flux.

consiste à pouvoir construire à la volée une partie du réseau d'accointances des agents : cela permet d'établir un flux d'agents dans lequel les documents circulent ce qui permet aux agents de les annoter à la volée (figure 1). Pour autant, il ne s'agit pas de fixer définitivement les communications. Celles-ci peuvent être modifiées au cours du temps et, de plus, l'agent dispose toujours du moyen de communication non figé qui est la communication classique par messages (définis par la FIPA et implémentés dans MadKit).

L'agent dispose donc de deux moyens de communications possibles :

1. Les documents qui circulent sur les entrées/sorties (leur nombre est déterminé par l'implémentation de l'agent : un agent qui filtre un document par mot-clés va par exemple posséder une seule entrée et deux sorties qui correspondent aux filtrages positif et négatif du document).
2. Les autres messages (broadcasts ou bien ciblés) qui restent utilisables (un exemple est donné dans le paragraphe 3 concernant l'agent de clustering).

Pour résumer, cette plate-forme permet la création de chaînes de traitements[3] intelligentes basées sur une approche multi-agents. Une description plus détaillée est donnée dans [Xavier Denis, 2002].

2.5 Capture d'écran

La figure 2 représente une capture d'écran de la plate-forme avec une chaîne de traîtement simple constituée d'un agent de dump, de quelques agents de filtrage (nouveauté, mots-clés) et d'un agent de classement par contenu (clustering). On peut aussi voir quatre clusters qui ont été créés. Le terme 'Macro Agent' designe un environnement fermé (d'où les messages ne peuvent sortir) dans lequel des agents évoluent ; plusieurs environnements peuvent évidemment exister simultanément dans la plate-forme.

[3]'Workflow' dans la littérature anglo-saxone

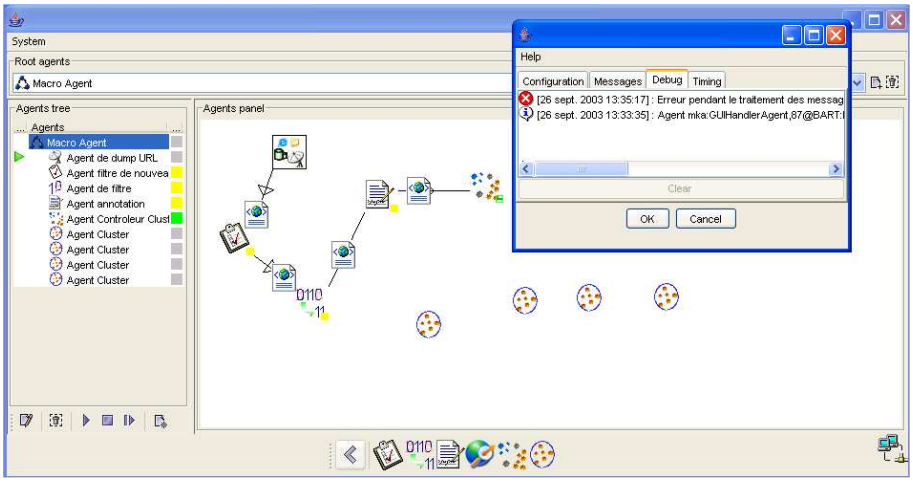

FIG. 2 – Capture d'écran de la plate-forme.

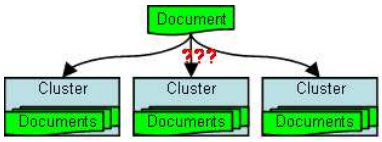

FIG. 3 – Problématique du clustering.

3 Le problème de la classification de documents

Un processus de veille implique souvent de devoir catégoriser des documents en fonction du centre d'interêt d'un utilisateur (figure 3). Cela se fait essentiellement par construction d'une bibliothèque d'apprentissage initiale représentant des ensembles de documents dits 'intéressants' et par l'ajout continuel de nouveaux documents dans ces catégories prédéfinies [Debole et Sebastiani, 2003]. Cependant, dans les approches classiques des logiciels de veille, il est très rare de pouvoir corriger des problèmes liés à la classification au cours du temps sans devoir remettre en cause le classement original.

Pour répondre à cette limitation, la plate-forme intègre un agent d'analyse chargé de classifier des documents arrivant sur son entrée de flux et qui gère le retour d'expérience de l'utilisateur. Cet agent est appelé 'contrôleur de clusters' (paragraphe 3.3) et utilise un autre type d'agents appelés 'cluster' (paragraphe 3.2) chargé de contenir les documents.

3.1 Operations disponibles sur les clusters

Afin de pouvoir améliorer les résultats du clustering au cours du temps, un certain nombre d'opérations doivent être disponibles pour l'utilisateur. Celles-ci sont décrites ci-dessous :

- Catégorisation automatique de nouveaux documents dans des clusters existants.
- Création automatique de nouveaux clusters pour des documents dont la thématique est éloignée de ceux présents dans les clusters.
- Ajout/suppression de documents dans des clusters.
- Suppression de clusters.
- Déplacement de documents d'un cluster à un autre.
- Apprentissage des besoins de l'utilisateur (éviter que des documents déplacés d'un cluster A à un cluster B ne reviennent dans A).
- Visualisation des documents d'un cluster.
- Affinage manuel des paramètres de sensibilité.
- Rejetter un document d'un cluster et le re-catégoriser immédiatement.

Toutes ces opérations sont disponibles dans l'implémentation actuelle de la plateforme. Une description de la façon dont les algorithmes fonctionnent est donnée dans le paragraphe 3.4.

3.2 Description des agents cluster

L'agent cluster est l'entité qui représente un cluster (ensemble de documents proches en terme de contenu). La proximité ou non de ces documents est définie par la mesure TFIDF communément utilisée en recherche d'information [Chakrabarti, 2000] et décrite au paragraphe 3.4. Le choix de cette mesure est avant tout liée à la simplicité et aux relativement bons résultats qu'elle obtient. Bien sûr il peut (et doit) exister plusieurs de ces agents pour représenter des groupes de documents semblables. Ainsi la problématique de catégorisation exposée dans la figure 3 consiste pour les agents cluster à analyser le contenu des nouveaux documents qu'ils recoivent et à répondre (positivement ou négativement) à l'évaluation par score en fonction des documents qu'ils contiennent déjà.

L'intérêt du clustering est de pouvoir identifier et regrouper des documents semblables. Cependant que se passe-t-il si un document est accepté par un cluster A alors qu'il devrait arriver dans un cluster B ? Et surtout comment modifier le système pour qu'un utilisateur puissse déplacer le document en question et indiquer à l'agent cluster A que des documents semblables ne doivent plus revenir ? De cette question est née le principe de liste noire de documents : chaque cluster contient sa liste de documents (dits positifs) et un ensemble de documents dits négatifs. Ainsi un agent cluster ne peut accepter un nouveau document que si celui-ci n'est pas trop proche (distance expliquée au paragraphe 3.4) des documents négatif. Ceux-ci se construisent au fur et à mesure des actions entreprises par l'utilisateur (ou par le système) : dès qu'un document d est déplacé de A vers B, d est automatiquement ajouté à la liste des documents négatifs du cluster A.

Ce paragraphe présente le principe de fonctionnement des agents clusters. Cependant, et pour permettre de décider quel agent cluster doit au final accueillir le document,

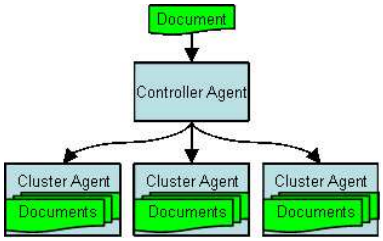

FIG. 4 – Problématique du clustering avec les agents.

un nouvel agent est nécessaire : cet agent organise un vote parmi les agents clusters et décide où va le document. Il est appelé agent contrôleur de clusters.

La figure 4 montre des agents cluster contenant des documents, l'agent contrôleur des clusters (paragraphe 3.3) et les documents entrants.

3.3 Description de l'agent contrôleur de clusters

Le principal but de l'agent contrôleur de clusters est de gérer les réponses données par les agents clusters lors de l'arrivée de nouveaux documents. Son fonctionnement est assez simple :

1. A la réception d'un document il contacte tous les agents clusters pour savoir combien existent.
2. S'il n'en existe aucun alors il en crée un et lui assigne le document. Sinon il interroge tous les clusters pour demander le score que chacun atttribue au document (en terme de distance représentée par la mesure TFIDF).
3. Il attend ensuite les réponses et sélectionne le cluster qui obtient le meilleur score pour lui assigner le document. Au cas où aucun agent cluster ne répond (car le document est trop proche de la liste noire des clusters), le document est associé à un nouveau cluster.

Il est important de préciser qu'un agent cluster doit connaître l'intégralité du corpus (ensemble des documents déjà catégorisés) pour qu'il puisse donner un score à un document (cela peut impliquer des problèmes d'espace mémoire qui ne sont pas pris en compte ici). Cette récupération du corpus est faite par l'agent contrôleur de cluster pendant la deuxième phase décrite précedemment :

1. Il envoit un message broadcast pour contacter les agents clusters. Ceux-ci répondent en envoyant un tableau contenant le vocabulaire (mots) de leurs documents.
2. A partir des réponses obtenues précédemment, il reconstruit le vocabulaire complet du corpus (étape nécessaire dépendante de l'implémentation de la mesure TFIDF) et envoit un nouveau message aux clusters qui ont répondu. Celui-ci contient le document à catégoriser et les informations sur le corpus. A partir de ce moment les agents clusters peuvent répondre en envoyant le score évalué.

$TFIDF(w_i, d) = tf(w_i, d) * idf(w_i)$
où $tf(w_i, d)$ représente la fréquence du mot w_i dans le document d
et $idf(w_i) = \log\left(\frac{1+N}{1+df(w_i)}\right)$ (N la taille maximale évaluée du corpus).

TAB. 1 – Mesure TFIDF

Document d et cluster C à évaluer
$$dist(d, C) = \frac{\sum TFIDF(w,d).TFIDF(w,C)}{\sqrt{\sum TFIDF(w,d)}.\sqrt{\sum TFIDF(w,C)}} \quad \forall w \in d \cap C$$

TAB. 2 – Distance cosine entre un document et un cluster

Cette étape permet de s'assurer que le corpus est bien synchronisé avec les agents clusters existants (en particulier si un utilisateur supprime des agents clusters à la main).

3.4 Description des algorithmes

Ce paragraphe concerne la description des algorithmes utilisés pour la classification des documents et un rappel de la mesure TFIDF. Celle-ci repose sur une représentation vectorielle des documents. En effet, chaque document va être représenté par un vecteur contenant la fréquence des mots dans l'espace vectoriel de l'ensemble du corpus[4] [Mladenic, 1998]. De la même façon un corpus est représenté par son centroïde qui est le vecteur représentant la fréquence de mots du cluster dans l'espace vectoriel du corpus complet.

La mesure TFIDF est depuis longtemps utilisée dans les problèmes de catégorisation : bien qu'assez simple, elle donne de bons résultats [Joachims, 1997]. La table 1 en donne la formule. La table 2 donne la distance entre un cluster et un document. Enfin la table 3 résume l'algorithme appliqué par l'agent contrôleur de clusters pour catégoriser un nouveau document. Il introduit un seuil de sensibilité de création de nouveaux clusters qui est à déterminer expérimentalement (en fonction des besoins de l'utilisateur et des types de documents).

3.5 Apport des agents dans la résolution du problème

Le problème tel qu'il a été décrit pour le moment est très proche d'une implémentation parallèlisée du clustering. Quel est donc l'apport du système multi-agents ?

Tout d'abord, il autorise une distribution du problème sur plusieurs ordinateurs : cela permet d'alléger la charge de traitement du logiciel. Ensuite, l'autonomie des agents ouvre la voie à un certain nombre de possibilités (non implémentées dans la version actuelle de la plate-forme mais en cours d'élaboration) :

[4] représentation par 'bag of words'

```
d : le nouveau document à catégoriser
clusterAgents = contacterTousLesAgentsClusters()
statsCorpus = calculerCorpus(clusterAgents)
Pour chaque agent dans clusterAgents
 scores += recueillirScore(agent)
Fin pour
Si scores est vide Alors
 créerNouveauCluster()
Sinon
 Si meilleurScore(scores) > seuil Alors
  AjouterAuMeilleurAgentCluster()
 Sinon
  créerNouveauCluster()
 Fin si
Fin si
```

TAB. 3 – Algorithme de décision de classification

- Réorganisation automatique des clusters par communication entre agents clusters : les agents peuvent s'échanger des documents qu'ils considèrent trop éloignés et les proposer à d'autres clusters (par un système de vote équivalent à celui décrit au paragraphe 3.4 mais sans passer par l'agent controleur de clusters). Bien que cette méthode ne soit pas encore implémentée, elle offre des perspectives tout à fait intéressantes surtout du point de vue de l'amélioration des résultats au cours du temps.
- Réorganisation automatique du flux d'agents par insertion d'agents filtrants (en fonction du retour d'expérience utilisateur). Ce filtre permettrait d'éliminer des documents dès leur téléchargement en fonction d'un retour d'expérience utilisateur plus général (refus explicite de documents).

4 Tests de performance

Ce paragraphe concerne les tests préliminaires de performance sur le clustering avec retour d'expérience.

4.1 Documents utilisés

Les tests ont été effectués sur une base de 30 documents issus du site http ://fr.news.yahoo.com. Bien sûr, cela ne sert pas à évaluer les performances brutes de la mesure TFIDF puisque de nombreux articles en parlent déjà. En fait, ces documents servent à confirmer que l'utilisation d'une liste noire peut améliorer grandement les performances de la classification et donc de la veille. Le tableau 4 décrit les sujets des documents récupérés.

Catégorie	Irak	Multimédia	Sports	Total
-	14	10	7	31

TAB. 4 – Sujets des documents de test.

N°	Taille	Sujet
1	14	14 Irak (5 Irak, 2 Kurdistan, 7 NU)
2	6	6 Multimedia (Microsoft, RedHat, HP)
3	1	1 Multimedia (intrusion)
4	1	1 Multimedia (mobilité)
5	1	1 Multimedia (blu-ray)
6	1	1 Multimedia (Linux XBox)
7	2	2 Sport (cyclisme)
8	1	1 Sport (rugby)
9	3	3 Sport (2 golf, 1 tennis)
10	1	1 Sport (basket-ball)

TAB. 5 – Clusters trouvés (sans retour d'expérience).

4.2 Tests

La première étape de la clusterisation consiste à trouver un seuil convenable pour le repérage de nouveaux clusters. Pour les documents présentés précédemment, celle-ci correspond à 0,15. Ainsi, les clusters obtenus automatiquement sont décrits dans la table 5. On peut voir qu'il existe deux clusters qui concernent Microsoft (clusters 2 et 6) : que se passe-t-il si de nouveaux documents concernant Windows doivent être catégorisés ? La table 6 répond à cette question en comparant les clusters choisis pour stocker de nouveaux documents issus du même site (5 concernant Windows) sans retour d'expérience de l'utilisateur avec ensuite avec (seuls les clusters modifiés 2 et 6 sont représentés). On peut voir que sans avoir déplacé le document parlant de Microsoft du cluster 2 au cluster 6, les nouveaux documents auraient été scindés en deux parties. La gestion de la liste noire permet donc d'empêcher qu'un document ne revienne dans un cluster.

Bien sûr, l'exemple des cinq documents ne sert ici qu'à illustrer le bon fonctionnement du principe de liste noire. D'autres tests ont été effectués et montrent que le retour d'expérience de l'utilisateur permet de construire des clusters mieux structurés au cours du temps. Il reste cependant à tester cette méthode sur un corpus plus grand et normalisé pour vérifier que la liste noire ne bloque pas entièrement le cluster.

5 Conclusion

Les tests effectués montrent qu'un retour d'expérience utilisateur peut grandement améliorer les performances d'un système de veille : la plate-forme présentée permet

N°	Taille	Sujet	Avant	Action	Après
2	6	6 (Microsoft, RH, HP)	+4 Windows	-1* Windows	
6	1	1 (Linux XBox)	+1 Windows	+1* Windows	+5 Windows

TAB. 6 – Clusters trouvés en déplaçant le document concernant Windows du cluster 2 vers le 6 (colonne 'Action').

d'utiliser des agents et ce retour d'expérience de manière assez simple. Cependant des tests plus poussés doient être réalisés pour confirmer ces résultats préliminaires (utilisation d'une base de données de documents reconnue) et une autre implémenation de l'algorithme de clustering doit aussi être choisie (SVM semble être une des meilleures possibilités et est en cours d'implémentation).

Références

[Brew, 2000] Chris Brew. Xml and linguistic annotation, 2000.

[Chakrabarti, 2000] Chakrabarti. Data mining for hypertext : A tutorial survey. *SIGKDD : SIGKDD Explorations : Newsletter of the Special Interest Group (SIG) on Knowledge Discovery and Data Mining, ACM*, 1, 2000.

[Debole et Sebastiani, 2003] Franca Debole et Fabrizio Sebastiani. Supervised term weighting for automated text categorization, 2003.

[Ferber et Gutknecht, 1998] J. Ferber et O. Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems, 1998.

[Goujon, 2000] B. Goujon. Extraction d'informations techniques pour la veille par exploration de notions indépendantes d'un domaine. *Terminologies nouvelles*, 19 :33–42, 2000.

[Gutknecht et Ferber, 2000] Olivier Gutknecht et Jacques Ferber. The MADKIT agent platform architecture. In *Agents Workshop on Infrastructure for Multi-Agent Systems*, pages 48–55, 2000.

[Joachims, 1997] Thorsten Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In Douglas H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 143–151, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

[Landau *et al.*, 1998] David Landau, Ronen Feldman, Yonatan Aumann, Moshe Fresko, Yehuda Lindell, Orly Liphstat, et Oren Zamir. Textvis : An integrated visual environment for text mining. In *Principles of Data Mining and Knowledge Discovery*, pages 56–64, 1998.

[McKelvie *et al.*, 1997] D. McKelvie, C. Brew, et H. Thompson. Using sgml as a basis for data-intensive nlp, 1997.

[Mladenic, 1998] Dunja Mladenic. Feature subset selection in text-learning. In *European Conference on Machine Learning*, pages 95–100, 1998.

[Sperberg-McQueen et Burnard, 2002] C.M.. Sperberg-McQueen et L. (eds.) Burnard. Tei p4 : Guidelines for electronic text encoding and interchange. text encoding initiative consortium., 2002.

[Woodruff *et al.*, 1996] Allison Woodruff, Paul M. Aoki, Eric Brewer, Paul Gauthier, et Lawrence A. Rowe. An investigation of documents from the World Wide Web. *Computer Networks and ISDN Systems*, 28(7–11) :963–980, 1996.

[Xavier Denis, 2002] Nicolas Chanchevrier Xavier Denis. Plate-forme de veille multi-approche pour l'aide à la décision. In *1ère JESIADIS*, pages 103–118, Brest, FR, 2002.

Summary

Many techniques exist to classify text documents depending on the user's needs (kNN, SVM, ...). Unfortunately, integrating these algorithms into a textmining platform is often done statically and prevents enhancing the processing in real-time. The aim of this article is to introduce a textmining platform in which heterogeneous data are represented using XML/TEI technology and where the user can interact with the full process of getting and analysing these data. To achieve that, the processing modules are represented as agents working on the MadKit platform. The learning process is based on a method derived from VSM and TFIDF using a kind of weighted black list (enabling undesired documents detection). The dynamic of the platform mainly relies on the possibility to add agents on the fly and to modify the parameters and order of the documents processing.