



550, rue Sherbrooke Ouest, bureau 100
Montréal (Québec) H3A 1B9
Tél. : (514) 840-1234; Téléc. : (514) 840-1244
888, rue St-Jean, bureau 555
Québec (Québec) G1R 5H6
Tél. : (418) 648-8080; téléc. : (418) 648-8141
<http://www.crim.ca>

Étude des frameworks UIMA, GATE et OpenNLP

Mustapha Es-salihe
Conseiller

Stéphane Bond
Agent de recherche senior

Développement et technologies Internet

29 mars 2006

TABLE DES MATIÈRES

TABLE DES MATIÈRES.....	2
LISTE DES TABLEAUX.....	3
LISTE DES FIGURES	4
INTRODUCTION	5
1. UIMA	6
1.1 Éléments de base de UIMA.....	6
1.1.1 Analysis Engines.....	6
1.1.2 Common Analysis Structure.....	6
1.1.3 Description des composants dans UIMA	7
1.2 Développement d'applications avec UIMA.....	7
2. GATE	8
2.1 Éléments de base de GATE	8
2.1.1 Language Ressource (LR)	8
2.1.2 Processing Ressources (PR)	9
2.1.3 Application.....	9
2.1.4 Visual Ressource	9
2.1.5 Plugins (CREOLE)	9
2.2 Développement d'applications avec GATE.....	9
3. OPENNLP	10
4. ÉVALUATION	12

LISTE DES TABLEAUX

Tableau 1 : comparaison des trois projets UIMA, GATE et OpenNLP 12

LISTE DES FIGURES

<i>Figure 1 : Fonctionnement des applications sous UIMA.....</i>	<i>8</i>
<i>Figure 2 : fonctionnement des applications avec GATE.....</i>	<i>10</i>

INTRODUCTION

L'information non structurée représente la source d'information largement disponible dans toutes les organisations, cette source est en constante évolution. L'immense quantité d'informations disponibles en plusieurs langues et en différents formats incluant le texte, l'audio, l'image et la vidéo dispersées partout constitue un contenu d'une importante valeur ajoutée, mais la nature non structurée de ce contenu le rend difficilement exploitable. La recherche et l'exploration des données à partir du contenu non structuré présente des défis majeurs.

Une application de gestion de l'information non structurée (UIM, Unstructured Information Management) permet d'analyser un grand volume d'informations non structurées afin de découvrir, organiser et diffuser des connaissances. Les résultats de cette analyse doivent être organisés dans des sources de données structurées pour permettre à des outils de recherche et d'exploration de données de les exploiter. Les applications UIM peuvent utiliser une variété de technologies dont principalement :

- Traitement de la langue naturelle
- Recherche d'informations
- Apprentissage machine
- Les ontologies
- Raisonnement automatique
- Reconnaissance de la parole

Ces technologies ont été développées séparément dans différents laboratoires et instituts de recherche en utilisant des techniques, des langages et des plateformes différents. Établir un pont entre le monde non structuré et le monde structuré à l'aide de ces technologies et dans un même environnement nécessite leur intégration, une opération très coûteuse.

Nous avons assisté dernièrement à la naissance de quelques frameworks qui ont comme objectif la composition et l'agrégation de plusieurs technologies d'analyse d'informations non structurées. Parmi ces frameworks, nous trouvons UIMA (Unstructured Information Management Architecture) d'IBM qui vient de le rendre *open source*, GATE (General Architecture of Text Engineering) et OpenNLP (Open Natural Language Processing).

L'objet de ce document est de décrire brièvement ces trois frameworks, de les comparer et surtout de présenter les intégrations possibles entre eux afin de maximiser la réutilisation des techniques et des solutions déjà en place pour l'analyse des informations non structurées.

1. UIMA

UIMA (<http://www.research.ibm.com/UIMA/>, <http://uima-framework.sourceforge.net/>) est une architecture logicielle qui spécifie les interfaces de composants, les structures de données, les patterns de conception et la démarche de développement pour créer, décrire, détecter, agréger et déployer des capacités d'analyse multimodale.

UIMA est aussi un framework qui fournit l'environnement d'exécution dans lequel les développeurs peuvent intégrer leurs implémentations de composants UIMA pour construire et déployer des applications.

UIMA fournit un kit de développement qui inclut une implémentation Java de l'architecture et du framework UIMA. Ce kit inclut aussi un ensemble de plug-ins Eclipse pour faciliter le développement avec UIMA.

1.1 Éléments de base de UIMA

Les éléments suivants composent les éléments de base du framework et de l'architecture UIMA.

1.1.1 Analysis Engines

Les *Analysis Engines* (AE) sont des composants de base utilisés pour analyser des documents afin de détecter des attributs descriptifs sous forme de méta données, appelés *Analysis Results* (AR). Un *document* dans UIMA est une unité de contenu qui peut contenir soit du texte, de l'audio ou de la vidéo. Les AR peuvent inclure des énoncés décrivant des régions d'une façon plus granulaire que le document source. Un *span* désigne une séquence de caractères dans un document. Un AE détecte des entités appelées *Types*, comme des personnes, des sujets, des villes, etc. Un exemple d'énoncé qu'un AE peut détecter dans un document peut être comme suit :

Le span de la position 101 à 112 dans le document D102 représente une Personne

Les AE sont des objets agrégés et gérés par le framework UIMA. Les développeurs développent des algorithmes d'analyses dans des composants appelés *Annotators* qui sont encapsulés dans les AE pour ajouter les interfaces et l'infrastructure nécessaires pour leur composition et leur déploiement dans le framework UIMA.

1.1.2 Common Analysis Structure

Common Analysis Structure (CAS) permet de représenter et partager les résultats d'analyse entre les *Annotators*, il s'agit d'une structure de données pour représenter des objets, des propriétés et des valeurs. UIMA fournit des types de base pour les utiliser dans une CAS et qui peuvent être étendus par les développeurs pour aboutir à un schéma plus riche de types, appelé *Type System* (TS). Un TS est spécifique à un domaine ou une application, et les types dans un TS peuvent être organisés dans une taxonomie.

UIMA supporte l'analyse simultanée de plusieurs vues du même document dans le but d'effectuer une analyse multimodale, comme par exemple une vue pour analyser l'audio et l'autre vue pour

analyser le sous titrage du même document (bande audio). Dans UIMA, une vue s'appelle *Subject of analysis* (Sofa). Une CAS contient dans ce cas un ou plusieurs Sofa avec les objets représentant les résultats d'analyse pour chaque Sofa.

1.1.3 Description des composants dans UIMA

UIMA contient un ensemble de composants de base dont les AE et les *Annotators* font partie. Les développeurs créent des composants en réutilisant les composants de base et/ou d'autres sous composants développés par des tiers. Deux parties sont essentielles pour mettre en place tout composant dans UIMA :

- La partie déclarative : qui décrit l'identité, la structure et le comportement du composant à l'aide de XML.
- La partie implémentation du composant à l'aide d'un langage de programmation comme Java

C'est une grande force d'UIMA car il s'agit d'une approche qui favorise la réutilisation et l'agrégation de composants. En se basant sur la description XML des composants, on peut effectuer une recherche de composants et une agrégation selon les critères définis.

Un AE primitif contient un simple *Annotator* qui effectue une tâche granulaire comme la détection de la langue, la tokenisation, la reconnaissance de phonèmes, etc. On peut agréger plusieurs AE primitifs pour créer un AE complexe qui permet d'effectuer une analyse plus complète comme la détection des entités nommées. Un AE agrégé inclut dans sa description le flux des composants qu'il utilise, le framework UIMA s'occupe de leur fournir l'environnement d'exécution selon l'ordre défini.

1.2 Développement d'applications avec UIMA

La figure 1 montre une vue d'ensemble du fonctionnement des applications qui utilisent UIMA.

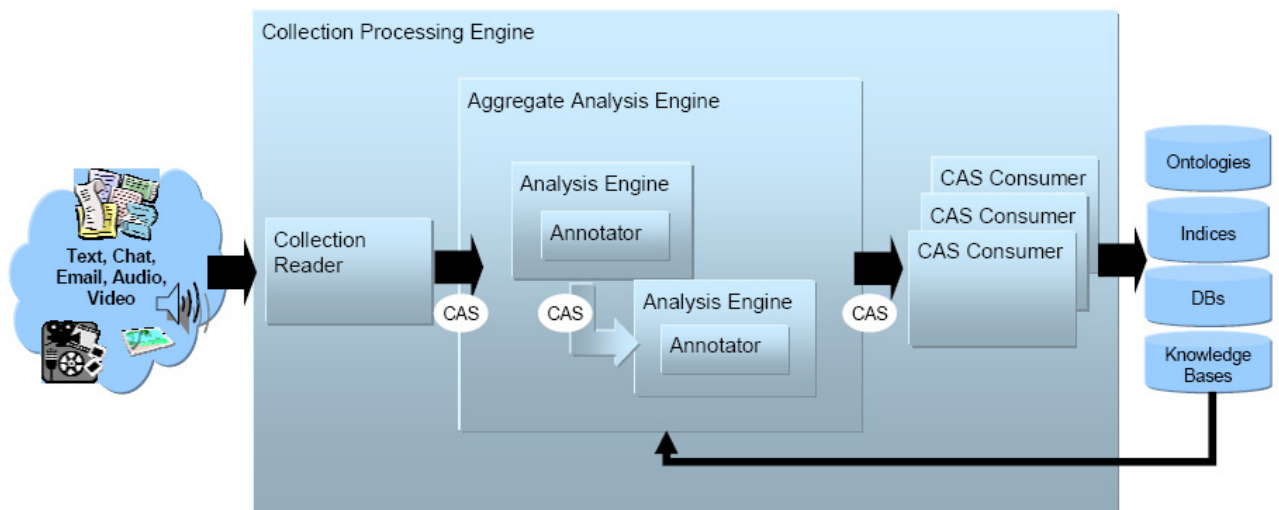


Figure 1 : Fonctionnement des applications sous UIMA

En plus des composants de base, d'autres composants sont définis dans UIMA pour supporter tout le cycle de développement des applications. Les principaux sont :

- *Collection Reader* : se connecte à une source d'informations et itère sur la collection des documents qui la compose pour initialiser les objets CAS avant l'analyse.
- *CAS Initialiser* : utilisé par UIMA pour accéder aux documents dépendamment de leur format physique, la tâche d'un *CAS Initialiser* est spécifique au format du document source et la logique du mappage de ce format vers les objets CAS.
- *CAS Consumer* : intervient à la fin du processus pour traiter les résultats d'analyse effectués par les différents AE et les rendre exploitables par d'autres applications, comme l'indexation des résultats pour un engin de recherche, le stockage dans une base de données, etc.
- *Collection Processing Engine (CPE)* : c'est un composant agrégat qui spécifie tous les composants qui participe dans le cycle de traitement d'informations dans UIMA partant d'un *Collection Reader*, passant par un ensemble d'AE jusqu'aux *CA Consumers*. Chaque CPE est décrit dans un fichier XML.
- *Collection Processing Manager (CPM)* : c'est un composant qui se base sur les descripteurs XML des CPE pour les déployer et les exécuter dans un environnement UIMA. Le CPM s'occupe de gérer les objets CAS, la distribution des composants, la reprise en cas d'incident et la gestion des performances.

2. GATE

GATE (<http://gate.ac.uk/>) est une infrastructure permettant le développement et le déploiement de composants pour le traitement de la langue naturelle. Le projet est constitué d'une architecture, d'un framework et d'un environnement de développement (IDE). L'architecture décrit les composants qui entrent dans le système de traitement de la langue naturelle. Le framework est utilisé comme base pour le développement des composants. Il est constitué de bibliothèques et de méthodes d'interactions. L'environnement de développement est une application autonome qui permet d'exécuter et de tester les composants et de modifier rapidement les paramètres de l'application.

2.1 Éléments de base de GATE

2.1.1 Language Ressource (LR)

Comprend toutes les sources de données statiques comme les lexiques, les taxonomies, les ontologies et autres ressources pouvant être exploités par les composants. Les documents traités et les collections, nommées corpus, peuvent aussi être définis comme Language Ressource. GATE et ses composants offrent le support pour plusieurs formats dont XML, RTF, HTML et SGML.

2.1.2 Processing Ressources (PR)

Représentent les algorithmes et composants effectuant un traitement. Il peut s'agir de parseurs, de convertisseurs, de moteurs d'analyses ou de tout autre élément ayant pour but d'ajouter ou de transformer des données. Lors de leur exécution, les PRs communiquent entre eux à l'aide d'objets. Ils peuvent ajouter de l'information aux documents traités sous forme d'annotations.

2.1.3 Application

Une application GATE aussi appelée contrôleur est une agrégation de plusieurs Processing Ressources sous forme de pipeline. Les pipelines peuvent être séquentiels ou conditionnels permettant ainsi de contrôler le flux d'exécution en fonction des données traitées. L'application prend en entrée un corpus de documents et produit des résultats sous forme visuelle ou dans un format de sortie spécifié.

2.1.4 Visual Ressource

Les Visual Ressources permettent la présentation des résultats à l'intérieur de l'environnement de développement GATE. En plus de la présentation des données et annotations recueillies, des éditeurs permettent la saisie de nouvelles informations par l'utilisateur. Ces informations pourront, par la suite, être reprises par les Processing Ressources. GATE permet l'ajout de Visual Ressources personnalisés permettant une adaptation en fonction des analyses effectuées.

2.1.5 Plugins (CREOLE)

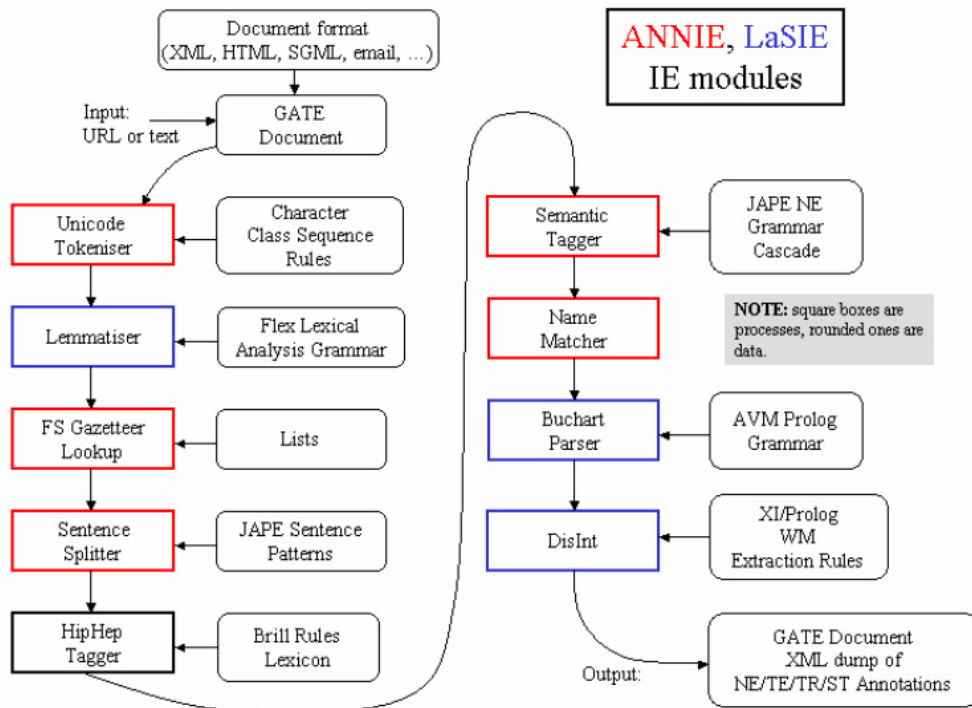
Les plugins peuvent être composés de plusieurs Language Ressources et Processing Ressources spécialisés dans l'exécution d'une tâche précise. Le module CREOLE, pour Collection of REusable Objects for Language Engineering, est livré avec le système de base et offre un large inventaire de moteurs d'analyses.

2.1.5.1 ANNIE

Ce plugin offre toute la gamme de Processing Ressources nécessaires au dépistage d'information sur du texte (Information Extraction). Il offre entre autres les outils pour le traitement de phrases, pour la détection des entités et pour la détection de références entre les sections d'un texte.

2.2 Développement d'applications avec GATE

Une application GATE se traduit par un pipeline par lequel est exécutée une série d'opérations d'analyses via les Processing Ressources.



Référence : <http://gate.ac.uk/sale/tao/>

Figure 2 : fonctionnement des applications avec GATE

3. OPENNLP

OpenNLP (<http://opennlp.sourceforge.net/>) est une structure de projets *open source* liés au traitement de la langue naturelle. L'objectif essentiel est de promouvoir et de faciliter la collaboration entre chercheurs et développeurs autour de tels projets. OpenNLP agit comme un parapluie pour l'ensemble de projets hébergés pour leur donner plus de notoriété et augmenter l'interopérabilité. Il s'agit aussi d'une initiative inclusive qui vise à intégrer tous les efforts de développement d'outils *open source* de traitement de langue naturelle. Tous les spécialistes et développeurs du domaine sont invités à verser leurs contributions afin d'enrichir le projet.

OpenNLP accueille également une variété d'outils Java de traitement de langue naturelle qui effectuent la détection de phrases, la tokenization, le tagging, la composition de blocs, la détection des entités nommées et le coréférencement en se basant sur le produit *open source* OpenNLP Maxent spécialisé en apprentissage machine.

La liste des projets OpenNLP est la suivante :

- Maxent : une librairie Java pour travailler avec les modèles d'entropie maximale.

- OpenNLP CCG Library : une collection de composants de traitement de langue naturelle et d'outils d'analyse à l'aide de la grammaire catégorielle combinatoire.
- OpenNLP Tools : une collection d'outils de traitement de langue naturelle qui utilise Maxent pour enlever les ambiguïtés.
- WordFreak : un outil Java d'annotation linguistique.
- AGTK : une suite de composants logiciels pour développer des outils d'annotation linguistique et de traitement de séries chronologies.
- Arithmetic Coding : une librairie Java pour le codage arithmétique.
- ComLinToo : une suite d'outils Perl pour la linguistique computationnelle.
- Attribute-Logic Engine (ALE) : un système pour la programmation logique et l'analyse grammaticale.
- EDG : un système LISP pour développer et afficher des HPSG (Head-Driven Phrase Structure Grammar)
- Ellogon : un composant sous License LGPL de traitement de langue naturelle écrit en C, C++, Java, Tcl, Perl et Python.
- Emdros : un engin de base de données pour le texte analysé ou annoté.
- FreeLing : une suite *open source* d'analyseurs de langage.
- Leo : un projet pour fournir une architecture de définition des spécifications XML de grammaires pour différents analyseurs de langue naturelle et d'outils de conversion automatique de grammaires entre ces systèmes.
- LKB : un environnement de développement de grammaire et de lexique pour utilisation avec des formalismes linguistiques à base de contraintes.
- Mallet : un kit Java pour l'apprentissage machine de langages.
- MinorThird : une collection de classes Java pour le tri du texte, l'annotation du texte, et l'apprentissage afin d'extraire des entités et catégoriser le texte.
- Ngram Statistics Package : permet le comptage et la mesure des Ngrams dans le texte.
- NLTK : une librairie Python pour faciliter le traitement de langue naturelle.
- nlpFarm : une collection de bibliothèques, d'outils et de démos en traitement de langue naturelle.
- SenseRelate : implémente un algorithme pour enlever les ambiguïtés des sens de mots en utilisant WordNet::Similarity (voir en bas).

- Tiger API : une librairie qui permet aux programmeurs Java d'accéder à la structure de n'importe quel corpus sous forme de fichier XML (tiger-xml)
- Weka : une collection d'algorithmes d'apprentissage machine pour effectuer les tâches de data mining.
- Weta : L'environnement Waikato d'analyse de texte.
- WordNet::Similarity : fournit les mesures d'approximations sémantiques en utilisant WordNet.

4. ÉVALUATION

Les trois projets sont forts intéressants pour tout ce qui est traitement de différentes sources d'informations. Il faut noter cependant que GATE et OpenNLP sont spécialisés en traitement de langue naturelle en se basant sur des documents texte, alors que UIMA est plus général et capable de travailler sur toute source de document (audio, texte, vidéo, etc.).

Le tableau suivant résume les principales caractéristiques des trois projets ainsi que leur force et leur faiblesse :

	UIMA	GATE	OpenNLP
Documentation	Bonne	Bonne	Moyenne
Architecture	Bien définie	Bien définie	Absence d'architecture globale
Langages de programmation	Java, C++	Java	Java, Python, autres
Mesures de performance	Pas d'indication	Pas d'indication	Pas d'indication
Algorithmes et techniques implémentées	Moyenne	Très riche	Riche mais non intégré
Limitations	Aucune	Aucune	Aucune
Types de documents traités	Tout	Texte	Texte

Tableau 1 : comparaison des trois projets UIMA, GATE et OpenNLP

Mis à part OpenNLP qui peut être considéré comme une source pour retrouver des implémentations de techniques de traitement de langue naturelle et les intégrer dans nos applications, UIMA et GATE peuvent être vus comme des projets complémentaires. GATE a été développé dans un milieu académique et de recherche dans lequel beaucoup de techniques et d'algorithmes robustes ont été implémentés, alors qu'UIMA est issu plutôt du milieu industriel pour des déploiements distribués et à large échelle. Le choix entre UIMA et GATE dépend de la nature des documents à traiter, et de la facilité de travailler avec l'environnement de développement Eclipse dans le cas d'UIMA ou avec des interfaces autonomes de tout environnement de développement mais plus conviviales dans le cas de GATE.

Des travaux ont été initiés pour développer des mécanismes d'intégration entre les trois projets :

- Encapsulation des outils OpenNLP dans UIMA à l'aide des Annotators : les exemples sont fournis avec le kit de développement d'UIMA fourni par IBM (<http://uima.lti.cs.cmu.edu/opennlp.html>).

- Couche d'interopérabilité UIMA / GATE pour permettre à des plugins GATE de s'exécuter à l'intérieur d'UIMA et à des composants UIMA de s'exécuter à l'intérieur de GATE. Cette couche a été développée par l'équipe GATE à l'université Sheffield avec le support d'IBM. Elle a été intégrée complètement dans GATE. Cette intégration va permettre aux applications qui adoptent UIMA d'aller chercher les implémentations des différentes techniques et algorithmes et des outils de visualisation dans GATE, et aux applications qui adoptent GATE de profiter de la facilité de déploiement et du bon support offerts par UIMA.
- Intégration d'OpenNLP et de GATE par l'université d'Edinburgh.

La conclusion est qu'il est recommandé de suivre le développement et l'évolution d'UIMA et de GATE, et d'envisager les mécanismes d'intégration déjà en place entre ces deux projets pour tirer le maximum d'avantages de chacun d'eux.