

Introduction à la plateforme d'annotation uima@rcIn

Erwan Moreau

Université Paris 13 & UMR CNRS 7030
erwan.moreau@lipn.univ-paris13.fr

14 octobre 2010

Principaux objectifs (1)

- ▶ Mise à jour / amélioration de “l'utilisabilité” des outils de la plateforme Ogmios/Alvis
 - ▶ **TreeTagger, YaTeA**
 - ▶ permettre utilisation en “boîte noire”
- ▶ Jeter les bases d'une plateforme UIMA dédiée aux tâches étudiées par l'équipe (annotation sémantique)
 - ▶ à long terme : améliorer la **gestion/organisation des développements logiciels** de l'équipe
 - ▶ outils plus uniformes, compatibles, modulables, maintenance facilitée
 - ▶ **évolutivité**

Principaux objectifs (2)

- ▶ Proposer une nouvelle approche dans la conception d'un *Type System*
 - ▶ **générique** : permettre/faciliter la composition de composants dans une chaîne
 - ▶ transmission des données entre composants non homogènes
 - ▶ qui permet les **annotations concurrentes**

Secondairement :

- ▶ Bibliothèques “utilitaires” Java
- ▶ Possibilité de mise en place d'une politique de gestion logicielle
 - ▶ ... si les intéressés le souhaitent bien sûr !

UIMA en bref

- ▶ un *Framework* pour “encadrer” l’analyse de grands volumes de données non structurées
 - ▶ “analyse” \approx découverte de connaissances “utiles”
 - ▶ implémentation nettement orientée données textuelles
- Ensemble d’**outils** et (surtout) de **conventions**
 - ▶ But : partage, faciliter la communication inter-composants
 - ▶ Systèmes complexes avec composants indépendants mais composables
 - ⇒ Standardisation des composants + flexibilité
- ▶ Implémentation Apache UIMA en **Java**
 - ▶ open-source, développement rapide

Caractéristique : composants par encapsulation

- ▶ Appel à un programme externe pour la réalisation de la tâche
- Avantage : coût de développement moindre, outils connus
- Inconvénients nombreux, non conforme à philosophie UIMA
 - ▶ portabilité ↘
 - ▶ failles potentielles (entrées/sorties, gestion d'erreurs, multi-threading)
 - ▶ rupture du contrôle UIMA sur les composants
 - ▶ conséquences sur chaîne complexe
 - ▶ perte d'efficacité temps/espace
- ▶ Gestion de ces problèmes
 - ▶ Librairie d'appel "protégé" à un programme extérieur
 - ▶ possibilité de lire/écrire à la volée
 - ▶ Librairie de (ré)-alignement, conversion entre formats

Contexte

- ▶ Projet Quaero
 - ▶ WP 3.4 “Développement d’une nouvelle plateforme d’annotation”
- ▶ Casting :
 - ▶ Maître d’œuvre : Laurent Audibert
 - ▶ Travail préliminaire par Sondes Bannour
 - ▶ Aide du LINA (Univ Nantes), notamment Fabien Poulard
- ▶ Prototype quasi-“version beta” (phase tests utilisateurs)
 - ▶ Composants principaux terminés
 - ▶ Ajout des dernières fonctionnalités
 - ▶ Documentation en cours de rédaction
 - ▶ *Release* version stable très bientôt

Annotation Results for monde_diplo.txt.xmi in /home/erwan/wip/svn_uima/UIMAv00maq/erwan/uima-in-

Surenchères du Pentagone, pressions des « faucons » japonais Le dilemme nucléaire du président Barack Obama

Tandis que Moscou et Washington s'apprentent à signer un accord sur la limitation de leurs armements nucléaires stratégiques, les Etats-Unis se préparent à rendre publique leur nouvelle doctrine en la matière. Celle-ci ayant fait l'objet de pressions du Pentagone et des « faucons » japonais, elle devrait être bien éloignée de la vision exprimée il y a encore quelques mois par le président Barack Obama.

Par Selia S. Harrison

Une dizaine de mots éloquentes aurait suffi au président Barack Obama pour s'avancer vers l'obtention du prix Nobel de la paix et devenir, à la fois, le héros des militants du désarmement et la bête noire des fanatiques du nucléaire. Lorsqu'il a promis de renouveler et d'étendre les accords conclus avec la Russie sur le contrôle des armes nucléaires - connus sous le nom de traités de réduction des armes stratégiques (Strategic Arms Reduction Treaty, Start) -, qui diminueraient modestement les arsenaux des deux pays, ces derniers n'ont pas été surpris. Mais ces croyants acharnés se sont inquiétés lorsqu'il a déclaré, à Prague, le 5 avril 2009 : « Nous réduirons le rôle des armes nucléaires dans notre stratégie de défense nationale. » D'autant plus que le président venait d'entamer la très officielle analyse critique de la position nucléaire (Nuclear Posture Review, NPR) établie à

Legend

☐ Documen... ☒ Interpreta... ☐ Lemma ☐ PartOfSpe... ☐ Sentence
☐ Token

Select All

Deselect All

Hide Unselected

Click In Text to See Annotation Detail

Annotations

▼ Interpretation

▼ Interpretation ("fanatiques")

begin = 747

end = 757

```
componentId = fr.lipn.nlptools.uima.treetag
```

confidence = 0.307451993227005

typeId = Interpretation

```
▽ serie = FSArray
```

```
serie = PartOfSpeech ("fanatiques")
```

begin = 747

end = 757

```
componentId = fr.lipn.nlptools.uima.tre
```

confidence = 0.307451993227005

typeId = PartOfSpeech

value = ADI

```
serie = Lemma("fanatiques")
```

▼ Interpretation ("fanatiques")

begin = 747

end = 757

```
componentId = fr.lipn.plttools.uima.treetag
```

confidence = 0.6925479769706726

typeId = Interpretation

```
▼ serie = FSArray
```

```
serie = PartOfSpeech ("fanatiques")
```

begin = 747

end = 757

```
componentId = fr.lipn.plptools.uima.tre
```

confidence = 0.6925479769706726

typeId = PartOfSpeech

value = NOM

```
▷ serie = Lemma ("fanatiques")
```

Plan

Introduction

Le framework UIMA

Approche uima@rcIn

Implémentation

Exemple

Conclusion et perspectives

Le framework UIMA

Présentation, objectifs

Concepts généraux

Type System

Annotateurs

Outils standard UIMA

Synthèse

UIMA : Introduction

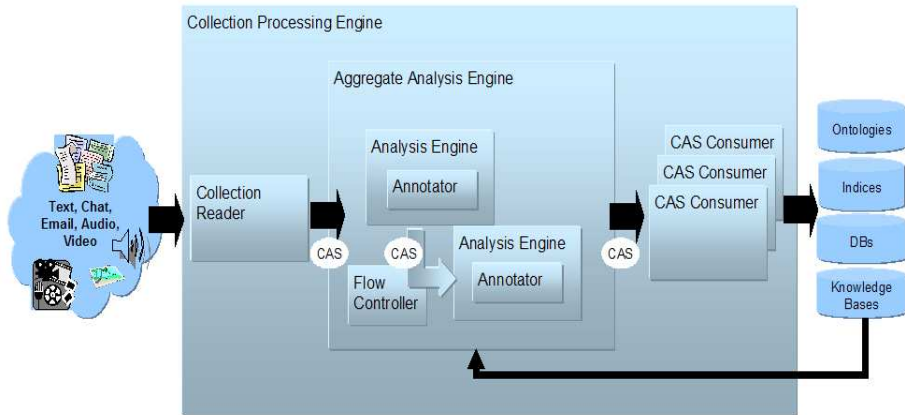
- ▶ Unstructured Information Management Applications :
 - ▶ **Spécification** définie par l'OASIS (Organization for the Advancement of Structured Information Standards)
 - ▶ **Implémentation** initiée par IBM, reprise par Apache (2006)
- ▶ But : encourager l'**interopérabilité entre composants**
 - ▶ *Représentation des données*, indépendance artefact / méta-données
 - ▶ *Modélisation/échanges de données* indépendants de la plateforme, forme "programmable"
 - ▶ *Découverte, ré-utilisation et composition* de composants indépendants
 - ▶ *Interopérabilité au niveau services*

Apache UIMA

- ▶ Une implémentation **Java** de la spécification UIMA OASIS
- ▶ Un *framework* + un *SDK* :
 - ▶ SDK : ensemble d'outils pour construire des composants UIMA via API
 - ▶ framework : librairies suffisantes pour déployer des composants
- ▶ Open-source
- ▶ Un projet Apache dynamique :
 - ▶ Version 2.3 \approx Janvier 2010, prochaine version bientôt
 - ▶ Apache "top-level project" depuis Mai 2010
 - ▶ Communauté d'utilisateurs grandissante (?)
 - ▶ Liste(s) d'utilisateurs active(s)

Principe

- ▶ Chaîne de traitement composée de :
 - ▶ un *Collection Reader*, qui lit le(s) document(s)
 - ▶ une séquence d'*Annotator Engine (AE)*, procédant chacun à une tâche donnée
 - ▶ un/plusieurs AEs qui “consomment” le résultat (ex- *CAS Consumers*)
- ▶ Le *CAS (Common Analysis Structure)* contient :
 - ▶ le document de départ (String)
 - ▶ les annotations (méta-données) ajoutées par les composants
 - ▶ le *Type System*
 - ▶ typologie pour la représentation des méta-données
- ▶ Gestion du processus par le framework via un *Collection Processing Engine (CPE)*
 - ▶ Passage de paramètres, gestion d'erreurs, “flot” (e.g. parallélisation), ressources, log...



Représentation des méta-données : le Type System

- ▶ Hiérarchie de types destinés à “recevoir” les annotations
- ▶ un type peut comporter des attributs (*features*)
 - ▶ ex : Individu avec features nom, prenom de type String
 - ▶ typage des features parmi types primitifs UIMA + tout type précédemment défini
- ▶ Héritage classique
 - ▶ ex : Chercheur hérite de Individu, feature h_index ajoutée.
- ▶ Annotation (instance d'un type) accédée via objet Java
- **MAIS** Type UIMA \neq objet Java
 - ▶ aucune méthode spécifique associée (accesseurs seuls)
 - ▶ interfaces/classes abstraites impossibles
- ▶ Type standard `org.apache.uima.jcas.tcas.Annotation`
 - ▶ Features begin/end de type int = position dans le texte

Construire un annotateur

- ▶ *Annotator Engine Descriptor* (fichier XML) :
 - ▶ Nom de la classe Java
 - ▶ Type System
 - ▶ Paramètres spécifiques
 - ▶ Nommage, typage, description, affectation
 - ▶ NB : entrée/sortie = CAS
 - ▶ *Capabilities* : types nécessaire en entrée et fournis en sortie
- ▶ Classe Java héritant de `JCasAnnotator_ImplBase`
 - ▶ Méthode `initialize(context)` : gestion des paramètres
 - ▶ Méthode `process(aJCas)` : tâche proprement dite
 - ▶ Accès au document : `aJCas.getDocumentText()`
 - ▶ Accès aux annotations avec itérateurs :
`aJCas.getAnnotationIndex(myType).iterator()`
 - ▶ Écriture d'annotations : `myAnnot = new MyType(aJCas) ...`
`myAnnot.addToIndexes()`

Outils (aperçu)

- ▶ API pour gestion d'erreurs, *logging*, etc.
- ▶ Composant *Collection Reader* qui lit les fichiers → CAS
- ▶ Composant *CAS consumer* qui écrit les données annotées en XML (format XMI)
- ▶ Outils graphiques
 - ▶ Costruction de descripteurs TS, AED (Eclipse)
 - ▶ Composition/configuration de CPE
 - ▶ Visualisation d'annotations
- ▶ Nombreuses façons d'utiliser/lancer un traitement :
 - ▶ CPE GUI (graphique)
 - ▶ script UIMA
 - ▶ programme Java

Le framework UIMA : synthèse

- ▶ Objectif essentiel : favoriser la **compositionnalité** entre composants d'analyse
- ▶ Moyen : **standardisation**
 - ▶ de la représentation des données
 - ▶ de "l'interfaçage" des composants (E/S, paramètres, etc.)
- ▶ Difficultés :
 - ▶ éviter de restreindre l'éventail des tâches représentables
 - ▶ rester \approx simple à utiliser (développeur/utilisateur)
 - ▶ efficacité
- ▶ Bénéfices secondaires :
 - ▶ Librairie d'outils standard
 - ▶ Accès aux composants extérieurs existants

Adopter UIMA ? (avis subjectif)

- ▶ Inconvénients :
 - ▶ Contraignant → construction + complexe, utilisation + laborieuse
 - ▶ Chronophage → apprentissage/adaptation, + de cas à gérer, - de marge de manœuvre
 - ▶ Rentabilité ? → moyen/long terme
 - ▶ Quand une “masse critique” de composants est disponible
 - ▶ Si ce standard reste dynamique
 - ▶ Avantages :
 - ▶ Standardisation : modularité, flexibilité, communication
 - ▶ Favorise programmation “propre”
 - ▶ sinon, UIMA inutile !
- ⇒ Bénéfice UIMA \approx langage non structuré vs. structuré
- ▶ contraignant d’abord, rentable ensuite
 - ▶ + robuste, débuggage simplifié, maintenance facilitée...

Approche uima@rcIn

Objectifs

Composants par encapsulation

Choix / guidelines

Type System

Objectifs

- ▶ Encapsuler les outils utilisés par l'équipe
 - ▶ TagEN (entités nommées), TreeTagger (POS), YaTeA (termes) (+LIA)
 - ▶ en faciliter l'utilisation (robustesse, flexibilité)
 - ▶ encodages, formats, problèmes d'alignement, etc.
- ▶ Plateforme **évolutive**
 - ▶ proposant des outils de base, dont
 - ▶ composants et applications immédiates
 - ▶ "utilitaires" de base (pour utilisation actuelle et composants futurs)
 - ▶ sur laquelle des composants futurs *non prévisibles* peuvent s'intégrer
 - ▶ éviter restrictions/contraintes → généricité
 - ▶ permettant l'usage d'annotations concurrentes

Différents utilisateurs

- ▶ **Boîte noire.** lancer un CPE (chaîne de traitement) prédéfini, sans contrôle (ou peu) sur les paramètres
- ▶ **Outils non UIMA.** utiliser les librairies indépendantes
- ▶ **UIMA end-user** configurer une chaîne de traitement faite de composants UIMA → pré-requis UIMA léger
- ▶ **Programmeur UIMA** construire de nouveaux annotateurs basés sur le même “cœur” → pré-requis UIMA+uima@rcIn
- ▶ **Maintenance** construire/améliorer la base des composants uima@rcIn → pré-requis UIMA+uima@rcIn sérieux !

Contraintes multiples

- ▶ Niveau inférieur → outils encapsulés
 - ▶ pas toujours bien spécifiés
 - ▶ pas toujours exempts de bugs
 - ▶ limitations intrinsèques
 - ▶ prévenir les problèmes liés à l'encapsulation
- ▶ Niveau supérieur → différents utilisateurs
 - ▶ facilité d'utilisation utilisateur occasionnel
 - ▶ liberté de paramétrage utilisateur avancé
 - ▶ API claire pour programmeur de composants
 - ▶ sans/peu contraintes pour composants futurs
- ▶ Framework UIMA
 - ▶ contraintes techniques/conceptuelles

Appel externe à un outil : difficultés

- ▶ **Portabilité** abandonnée.
 - ▶ à souligner pour les composants concernés car rare
- ▶ **Sûreté** du processus UIMA diminuée
 - ▶ rupture des mécanismes de contrôle (exceptions, log, ...)
 - ▶ failles potentielles du programme externe
- ▶ **Facilité d'emploi** diminuée
 - ▶ nécessité d'installer/localiser l'outil externe
 - ▶ parfois contraintes inattendues pour cadre UIMA
 - ▶ erreurs non filtrées du programme
- ▶ Transmission des **entrées/sorties**
 - ▶ conversions de formats (erreurs, pertes)
 - ▶ erreurs "physiques" (échec, blocage)
- ▶ **Efficacité**
 - ▶ perte en temps/espace pour transmission
 - ▶ surcharge mémoire (stockage en double)

Principe de précaution (bonnes pratiques)

- ▶ Programmation modulaire
 - ▶ Classes/packages dédiées à une tâche (ex : conversions)
 - ▶ préférer classes Java officielles
 - ▶ ré-utilisabilité du code testé
 - ▶ gestion harmonisée des cas/paramètres
 - ▶ inconvénient : plus complexe
 - ▶ *bug-free* impossible → faciliter débogage
- ▶ Documentation claire et complète
 - ▶ spécifications des composants, comportement particulier
 - ▶ choix d'implémentation
- ▶ Respect des conventions le cas échéant
 - ▶ faciliter reprise du code
 - ▶ ex : nommage de packages/classes, langue, etc.

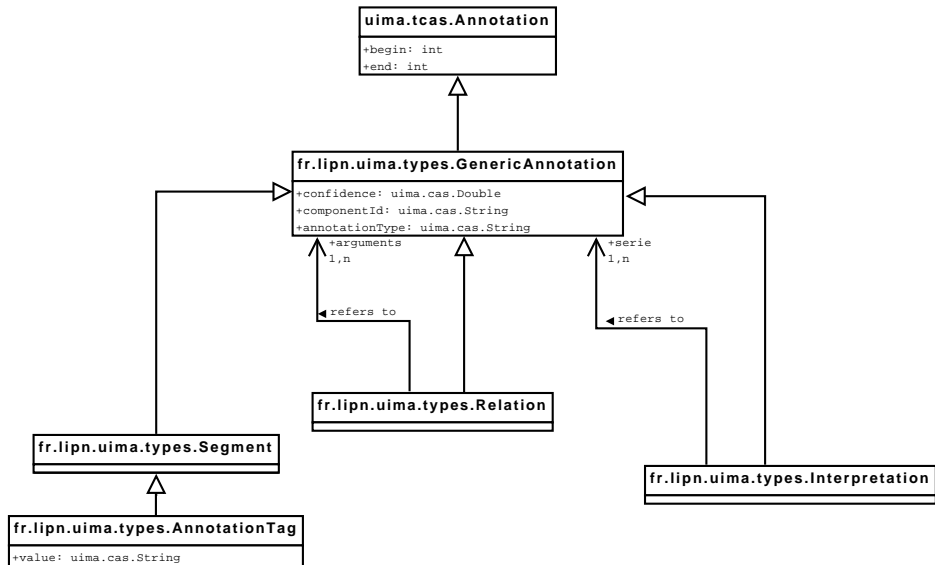
Orientation du Type System

- ▶ 2 grandes approches :
 - ▶ Précis, exhaustif
 - ▶ typologie riche, attributs prédéfinis
 - ▶ complexe mais simple à utiliser
 - cadre strict, besoins non prévus non/peu réalisables
 - ▶ Générique, abstrait.
 - ▶ flexible, modulable selon besoin
 - plus difficile à *bien* utiliser, moins “propre”

⇒ TS générique car flexibilité indispensable dans notre cas

- ▶ Ensemble restreint de types “minimaux”, “peu typés” :
 - ▶ facilite interprétation, *mapping* (couples attribut/valeur)
 - ▶ extensible localement
 - ▶ favorise combinaison d’annotations indépendantes, notamment concurrentes

LIPN Type System



Annotations concurrentes

- ▶ Ensemble distincts d'annotations, même texte, rôle similaire
 - ▶ ex1 : comparaison de différents outils sur la même tâche
 - ▶ ex2 : on veut annoter par 2 composants A et B ;
tokenization par A' meilleure pour A, mais par B' pour B
 - ▶ ex3 : ambiguïté locale entre 2 analyses syntaxiques possibles
- ▶ Problèmes :
 - ▶ Représentation dans le Type System ?
 - ▶ Annotateur "non conscient" des séries concurrentes ?
- ▶ Options envisagées :
 - ▶ Feature `componentId` pour distinguer l'annotateur créateur
 - ▶ Type Interpretation avec feature "liste d'annotations"
 - ▶ Concept des *vues* UIMA

Implémentation

Organisation générale

Module indépendant *lipn-nlptools-utils*

Composants UIMA *lipn-uima-core*

Vue d'ensemble

- ▶ Deux modules
 - ▶ *lipn-nlptools-utils* : packages non UIMA
 - ▶ `fr.lipn.nlptools.util.*`
 - ▶ appel programme externe + alignement / conversions format
 - ▶ *lipn-uima-core* : packages UIMA, dépend de *lipn-nlptools-utils*
 - ▶ `fr.lipn.nlptools.uima.*`
 - ▶ Composants TagEN, TreeTagger, LIA*, YaTeA + “boîte à outils”
- ▶ 2 “formes” pour *lipn-uima-core* :
 - ▶ Librairie JAR pour déploiement
 - ▶ Environnement avec scripts, doc, sources, tests,
- ▶ Conçu pour prise en main progressive
 - ▶ de *end-user* à développeur UIMA

Appel de programme externe

- ▶ Transmission “à la volée”
 - ▶ pas de double/triple occupation mémoire
 - ▶ temps réduit : lire/écrire simultanément < lire puis écrire
 - ▶ pas d'accès disque
- ▶ Difficultés
 - ▶ Erreurs d'E/S
 - ▶ Flux d'E/S : risque d'interblocage
 - ▶ appelé écrit sur stdout, attend lecture par appelant
 - programme appelant doit lire avant fin du programme !
 - ⇒ parallélisme, avec threads Java
 - ▶ risques habituels, garantir terminaison dans tous les cas
 - ▶ + spécificités UIMA : le CAS n'est pas “partageable”
- ▶ Objets Reader et Writer utilisés (simples, flexibles)

(Ré-)alignement, conversions

- ▶ Nombreuses conversions CAS \leftrightarrow format programme
- ▶ Modularité : 3 composants génériques
 - ▶ `AnnotatedTextReader` lit le texte annoté
 - ▶ `InputReader` reçoit chaque token + compare
 - ▶ `AlignerConsumer` consomme ces données
- ▶ Intérêts : combinaisons de composants, unicité du code
 - ▶ paramètres variés pour tous les cas
 - ▶ débuggage plus facile
- ▶ Objets Reader utilisés (simple, flexible)
- ▶ Formats “un token par ligne”, *ML (balises), positions

Boîte à outils

- ▶ Annotateur générique pour programme encapsulé
 - ▶ paramètres communs : langage, chemin, encodage, time out
 - ▶ environnement d'exécution, erreurs éventuelles
- ▶ Encapsulation d'itérateurs spécifiques :
 - ▶ synchronisation (si threads)
 - ▶ annotations souvent utilisées ensemble
 - ▶ Token, PartOfSpeech, Lemma (gestion superposition)
 - ▶ séries concurrentes :
 - ▶ lecture avec *contraintes* sur componentId
 - ▶ prise en compte des Interpretation
- ▶ Composants généraux :
 - ▶ Lecture/écriture du CAS au format "un token par ligne"
 - ▶ Autres utilitaires (prévus !)
 - ▶ ex : fusion/décomposition de documents

Composants encapsulés

Quelques principes :

- ▶ Conserver au maximum les fonctions du programme
 - ▶ paramètres, données en sortie
- ▶ Signaler les erreurs au plus tôt
 - ▶ éviter d'appeler le programme si paramètre non valide
 - ▶ vérifier les caractères spéciaux
- ▶ Utiliser les méthodes officielles
 - ▶ Ex : fichier temporaire, questions d'encodage, XML...
- ▶ Envisager les différentes utilisations
 - ▶ le code doit permettre la parallélisation
 - ▶ éviter les suppositions
 - ▶ ex : les documents ne sont pas nécessairement des fichiers

Exemple

Descripteur de Type System (Eclipse)

Descripteur d'AE (Eclipse)

Configuration avec le CPE GUI

Visualisation

Type System Definition

Types (or Classes)

The following types (classes) are defined in this analysis engine descriptor.

The grayed out items are imported or merged from other descriptors, and cannot be edited here. (To edit them, edit their source files).

Type Name or Feature Name	SuperType or Range	Element	
<input type="checkbox"/> fr.lipn.nlptools.uima.types.AnnotationTag	fr.lipn.nlptools.uima.types.Segment		<div>Add Type</div> <div>Add...</div> <div>Edit...</div> <div>Remove</div> <div>Export...</div> <div>JCasGen</div>
value	uima.cas.String		
fr.lipn.nlptools.uima.types.DocumentDivision	fr.lipn.nlptools.uima.types.Segment		
<input checked="" type="checkbox"/> fr.lipn.nlptools.uima.types.GenericAnnotation	uima.tcas.Annotation		
<input checked="" type="checkbox"/> fr.lipn.nlptools.uima.types.Interpretation	fr.lipn.nlptools.uima.types.GenericAnnotation		
fr.lipn.nlptools.uima.types.Lemma	fr.lipn.nlptools.uima.types.AnnotationTag		
fr.lipn.nlptools.uima.types.NamedEntity	fr.lipn.nlptools.uima.types.AnnotationTag		
fr.lipn.nlptools.uima.types.PartOfSpeech	fr.lipn.nlptools.uima.types.AnnotationTag		
<input type="checkbox"/> fr.lipn.nlptools.uima.types.Relation	fr.lipn.nlptools.uima.types.GenericAnnotation		

Imported Type Systems

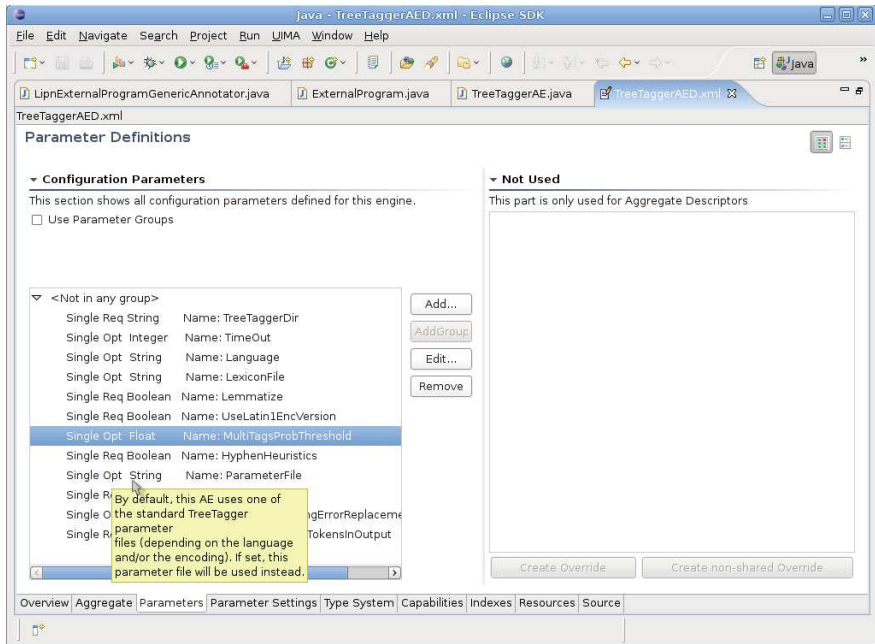
The following type systems are included as part of this one.

Add...

Remove

Set DataPath


Kind	Location/Name
By Name	fr.lipn.nlptools.uima.common.lipn-base-TS



le CPE GUI

Collection Processing Engine Configurator

File View Help

 **Unstructured Information Management Architecture**
An Apache Incubator Project

Descriptor: Browse..

Input Directory: Browse..

Encoding:

Language:

Analysis Engines

Add... << >>

☒ TreeTaggerTokenizerAED ☒ TreeTaggerAED ☒ YateaAED

Tree Tagger Dir: Browse.. Time Out:

Language:

Lexicon File:

Lemmatize: ☒

Use Latin1 Enc Version: ☐




Multi Tags Prob Threshold: Hyphen Heuristics: ☐

CAS Consumers

Add... << >>

☒ Xmi Writer CAS Consumer

Output Directory: Browse..

Initialized

Annotation Results for monde_diplo.txt.xml in /home/erwan/wip/svn_uima/UIMAv00maq/erwan/uima-in-

Surenchères du **Pentagone**, pressions des « faucons » **japonais**
Le dilemme nucléaire du président Barack Obama

Tandis que Moscou et Washington s'apprêtent à signer un accord sur la limitation de leurs armements **nucléaires** stratégiques, les Etats-Unis se préparent à rendre **publique** leur nouvelle **doctrine** en la matière. Celle-ci ayant fait l'objet de pressions du **Pentagone** et des « faucons » **japonais**, elle devrait être bien éloignée de la vision exprimée il y a encore quelques mois par le président Barack Obama.
Par Selig S. **Harrison**

Une dizaine de mots éloquents auront suffi au président Barack Obama **pour** s'avancer vers l'obtention du prix **Nobel** de la paix et devenir, à la fois, le héros des militants du désarmement et la **bête noire** des **fanatiques** du **nucléaire**. Lorsqu'il a promis de renouveler et d'étendre les accords conclus avec la Russie sur le contrôle des armes nucléaires - connus sous le nom de traités de réduction des armes stratégiques (Strategic Arms Reduction Treaty, Start) -, qui diminueraient modestement les arsenaux des deux pays, ces **derniers** n'ont pas été **surpris**. Mais ces **croayants** acharnés se sont inquiétés lorsqu'il a déclaré, à Prague, le 5 avril 2009 : « Nous réduirons le rôle des armes nucléaires dans notre stratégie de défense nationale. » D'autant plus que le président venait d'entamer la très officielle **analyse critique** de la position nucléaire (Nuclear Posture Review, NPR) établie à

Click In Text to See Annotation Detail

- Annotations
 - Interpretation
 - Interpretation ("fanatiques")
 - begin = 747
 - end = 757
 - componentId = fr.lipn.nlptools.uima.treetag
 - confidence = 0.307451993227005
 - typed = Interpretation
 - serie = FSArray
 - serie = PartOfSpeech ("fanatiques")
 - begin = 747
 - end = 757
 - componentId = fr.lipn.nlptools.uima.tre
 - confidence = 0.307451993227005
 - typed = PartOfSpeech
 - value = ADJ
 - serie = Lemma ("fanatiques")
 - Interpretation ("fanatiques")
 - begin = 747
 - end = 757
 - componentId = fr.lipn.nlptools.uima.treetag
 - confidence = 0.6925479769706726
 - typed = Interpretation
 - serie = FSArray
 - serie = PartOfSpeech ("fanatiques")
 - begin = 747
 - end = 757
 - componentId = fr.lipn.nlptools.uima.tre
 - confidence = 0.6925479769706726
 - typed = PartOfSpeech
 - value = NOM
 - serie = Lemma ("fanatiques")

Legend

☐ Documen...
☒ Interpreta...
☐ Lemma
☐ PartOfSpe...
☐ Sentence

☐ Token

Select All Deselect All Hide Unselected

Idées de composants futurs

- ▶ Composants :
 - ▶ Étiqueteur de termes
 - ▶ Segmenteur paramétrable
 - ▶ Analyse syntaxique (parser de Stanford,...)
- ▶ Intégration de composants UIMA extérieurs :
 - ▶ Dunamis, outil graphique de paramétrage/visualisation (LINA)
 - ▶ Nombreux composants LINA : lecture de pdf, identifieur de langue, extraction de termes, etc.
 - ▶ OpenNLP Wrapper : intégration des composants OpenNLP

Synthèse

- ▶ Contre : nombreux inconvénients !
 - ▶ Temps d'apprentissage/adaptation
 - ▶ problèmes fréquents de chemin/CLASSPATH
 - ▶ Cadre contraignant
 - ▶ nécessité de se documenter souvent
 - ▶ tests plus laborieux
- ▶ Pour : bénéfices à moyen/long terme
 - ▶ Combinaisons complexes de composants
 - ▶ Simplifier le remplacement d'un composant par un autre

⇒ **Utilité proportionnelle au niveau d'analyse**

- ▶ Plus il y a de phases antérieures, plus il est important :
 - ▶ d'avoir des composants fiables aux niveaux inférieurs
 - ▶ de pouvoir paramétrer simplement ces niveaux inférieurs

Stratégie de gestion des développements logiciels

- ▶ Intérêts d'une politique de gestion logicielle
 - ▶ Clarté des composants existants
 - ▶ éviter les doublons
 - ▶ harmoniser les approches
 - ▶ éviter perte de temps à chercher bonne version, doc...
 - ▶ Maintenance facilitée
 - ▶ centralisation des bugs/manques
 - ▶ trouver/corriger plus vite les problèmes
- ▶ Moyens nécessaires :
 - ▶ Dépôt centralisé (semi-)public
 - ▶ Identification des auteurs, versions, stabilité
- ▶ Outils libres existants
 - ▶ svn, maven,...
 - ▶ SourcesSup : plateforme de gestion de projets Enseignement Supérieur/ Recherche