Vers des outils robustes et interopérables pour le TAL : la piste UIMA

Fabien Poulard, Erwan Moreau, Laurent Audibert

Vers des outils robustes et interopérables pour le TAL : la piste UIMA

Fabien Poulard[1] Erwan Moreau[2] Laurent Audibert[2]
(1) Laboratoire d'Informatique de Nantes Atlantique (LINA) - UMR CNRS 6241
(2) Laboratoire d'Informatique de l'université Paris-Nord (LIPN) - UMR CNRS 7030
Fabien.Poulard@univ-nantes.fr, Erwan.Moreau@lipn.univ-paris13.fr,
Laurent.Audibert@lipn.univ-paris13.fr

Résumé. La complexification des tâches d'analyse en TAL (quantité et diversité de données, perfectionnement des techniques) offre de nouveaux défis en termes d'architecture logicielles ; pour y répondre, les systèmes d'analyse doivent devenir plus modulaires, interopérables et parallélisables. Dans cet article, nous montrons en quoi cette évolution est nécessaire à la poursuite des progrès en TAL, qui en tant que science expérimentale est très dépendante de la qualité de ses outils. Nous présentons ensuite UIMA, qui nous semble l'initiative la plus à même de rendre cette évolution possible.

Abstract. The growing complexity of analysis tasks in NLP (amount and variety of data, sophisticated techniques) offers new challenges in terms of software architecture ; thus the analysis systems must become more modular, interoperable and parallelizable. In this paper, we show how this evolution is a necessary condition for further progress in NLP, which as an experimental science highly relies on the quality of its tools. Then we present UIMA, which is in our opinion the most promising initiative.

Mots-clés : outils pour le TAL, interopérabilité, parallélisation des traitements, UIMA.

Keywords: tools for NLP, interoperability, processing parallelization, UIMA.

1 Introduction

Dans le « Jeopardy challenge », le super-ordinateur Watson d'IBM [1] parvient à gérer simultanément plusieurs tâches complexes liées au Traitement Automatique des Langues (TAL). Si cette prouesse technique doit beaucoup aux impressionnantes ressources matérielles, sa mise en œuvre n'a été rendue possible que grâce à l'utilisation d'une architecture logicielle robuste et performante : celle-ci permet à de multiples briques logicielles d'analyse de communiquer les unes avec les autres et de constituer ainsi des chaînes de traitement qui peuvent être déployées en parallèle.

Cet exemple médiatique d'actualité illustre à nos yeux l'évolution nécessaire des systèmes d'analyses en TAL vers la production d'analyses plus complexes sur des masses de données plus importantes. En effet, la complexité croissante des tâches à accomplir à des précisions toujours plus élevées s'accompagne d'une sophistication des techniques, d'un accroissement des volumes ainsi que d'une plus grande diversité des données traitées. Une plus grande modularisation de nos outils est nécessaire (plus de composants, des composants plus spécifiques). Ainsi les nombreux composants dépendent les uns des autres, et l'architecture hiérarchique classique (la sortie des composants de niveau inférieur constituant l'entrée de ceux de niveau supérieur) devient même parfois insuffisante : les composants peuvent avoir à fonctionner en parallèle et échanger des données à la volée.

Nous détaillons dans la section 2 les raisons pour lesquelles il nous semble nécessaire de replacer le développement d'outils robustes au cœur de la recherche en TAL, et pourquoi nous avons besoin de bases d'ingénierie solides pour y parvenir. Nous défendons cependant l'idée qu'il s'agit bien d'une problématique scientifique, et pas seulement d'ingénierie. Nous présentons ensuite dans la section 3 la spécification UIMA [2]. Nous discutons notamment de ses atouts qui en font un choix judicieux pour atteindre les objectifs suivants : interopérabilité des composants

1. http://www-03.ibm.com/innovation/us/watson/what-is-watson/the-future-of-watson.html
2. http://uima.apache.org

pour la construction de systèmes complexes, et parallélisation des chaînes de traitement pour traiter des masses de données importantes. La maturité d'UIMA n'est plus à démontrer, puisqu'elle constitue l'un des piliers logiciels du super-ordinateur Watson évoqué précédemment [3]. Finalement nous concluons sur la nécessité d'élargir les collaborations entre les différentes équipes de recherche autour du développement d'outils robustes.

2 Enjeux présents et à venir en TAL

2.1 Des outils fiables pour des expérimentations fiables

À l'instar des autres domaines scientifiques, le TAL avance par empilement de réussites successives : les progrès significatifs reposent sur la qualité des avancées établies précédemment (*standing on the shoulders of giants*). Cet empilement n'est pas seulement théorique mais également expérimental, puisque les aboutissements les plus complexes (résolution de références par ex.) reposent sur des traitements plus simples (découpage en mots et phrases, étiquetage des rôles grammaticaux, analyse syntaxique, etc.) considérés comme parvenus à un niveau de maturité suffisant. Ces empilements complexes correspondent à la notion bien connue de "chaînes de traitement" (la sortie d'un composant constituant l'entrée du suivant). Néanmoins ceci suppose que le coût d'adaptation d'un composant à un autre ne soit pas trop élevé : à cause de la profusion de modèles et de formats différents ainsi que la faible paramétrisation de nombreux composants, la tendance est au développement de solutions *ad-hoc* plutôt que par capitalisation de l'existant [4] ; si cela pouvait sembler rentable à court terme pour des composants simples (par ex. segmenteurs), il n'en sera pas de même avec des composants plus sophistiqués (par ex. analyseurs syntaxiques [5]).

Le principe des chaînes de traitement entraîne également la propagation des erreurs, dont on sait qu'une partie est inévitable lorsqu'on travaille sur le langage. Ainsi le succès et la validité des expérimentations scientifiques menées à un niveau donné dépend de la qualité et de la robustesse des composants utilisés en amont : il est donc souhaitable de pouvoir utiliser des composants de référence dont la fiabilité a été démontrée, sans être limité (ou le moins possible) par des questions de formats. Cette question se fait évidemment de plus en plus cruciale au fur et à mesure que l'on se place à un niveau d'analyse élevé, puisqu'alors la moindre erreur (évitable) qui se propage peut faire échouer l'expérimentation. Les méthodes expérimentales d'aujourd'hui sont les outils pour les expérimentations de demain. Il est primordial, une fois les méthodes validées, de consacrer du temps à leur finalisation afin d'en faire des outils dignes de ce nom et s'assurer de la qualité de nos expérimentations futures. À titre de comparaison, la physique ne serait pas ce qu'elle est sans les outils de détection de particules de G. Charpak, ce qui lui a d'ailleurs valu un Prix Nobel.

2.2 Complexification des systèmes et interopérabilité

Durant les dernières décennies, la plupart des problématiques majeures en TAL se déclinaient en problèmes relativement simples dans leur définition, au sens où il s'agissait certes de problèmes complexes mais dont le nombre et la nature des données étaient (relativement) fixés. Les grandes problématiques actuelles en TAL sont désormais plus difficiles à spécifier précisément en termes d'entrées/sorties, ce qu'on peut expliquer à travers deux grandes tendances :

— Les problématiques se situent généralement à un niveau d'analyse linguistique globalement plus élevé qu'auparavant, typiquement sémantique ou pragmatique. Or à de tels niveaux les choix de modélisation ou de représentation sont démultipliés : par exemple, on ne peut spécifier de façon stricte la nature et le nombre de paramètres à la problématique Question/Réponse en général. De plus ce type de tâche dépend très souvent d'analyses de niveau linguistique inférieur, ce qui complique bien sûr leur définition précise puisque deux systèmes de même rôle ne font pas nécessairement appel aux même outils ni à la même représentation des données.

— Conjointement, les ressources se sont multipliées en nombre et en forme. Une part importante des différences entre systèmes d'analyse de TAL récents ne concerne plus la technique mise en œuvre mais les

3. https://blogs.apache.org/foundation/entry/apache_innovation_bolsters_ibm_s

4. Rallongeant ainsi souvent la liste des formats incompatibles entre eux, et diminuant par conséquent les possibilités d'utilisation future de leur composants par d'autres.

5. Qui sont déjà partie intégrante de composants de prétraitement pour l'anglais.

ressources qui ont servi à sa mise au point ou qui servent directement à son fonctionnement. De nombreux paramètres caractérisent les ressources (et distinguent par conséquent les systèmes) :
— nature : corpus brut, pré-traité, analysé, vérifié manuellement, données structurées, etc. ;
— origine : créé spécifiquement, issu de pages internet, généré par des outils d'analyses, etc. ;
— forme (si données structurées) : modèle de représentation sous-jacent, jeux d'étiquettes, format d'échange ;
— langue ;
— volume.

Ainsi la complexité grandissante de ces tâches explique la difficulté à en donner une spécification formelle claire, et a des répercussions directes sur les possibilités d'échanges entre composants conçus indépendamment. Par conséquent il est très probable que de nombreux systèmes n'utilisent pas complètement les ressources existantes qui pourraient contribuer à leur qualité, à cause de la difficulté à appréhender leur diversité, ou plus précisément de la difficulté à communiquer avec les composants spécialisés qui en sont capables (ou qui produisent ces ressources). Cet éparpillement d'implémentations hétérogènes nuit à la progression du domaine :

— la reproductibilité des expériences menées avec un système donné est compromise, quand bien même une description claire en est fournie ;
— l'effort requis pour implémenter un système profitant des points forts de systèmes tiers est très coûteux ;
— la comparaison et l'évaluation des systèmes sont affectées, exception faite des campagnes d'évaluation (qui restent néanmoins coûteuses et restreignent l'évaluation à un jeu de données en particulier).

C'est pourquoi il est nécessaire que les systèmes d'analyse deviennent progressivement moins monolithiques et auto-suffisants, et au contraire capables de communiquer avec des éléments logiciels extérieurs. Ceci peut prendre différentes formes : de façon très directe, l'intégration d'un composant logiciel indépendant auquel une tâche particulière est déléguée ; ou l'appel à un composant totalement extérieur via une interface publique (par exemple de type service web) ; ou encore par l'utilisation de données statiques créées précédemment par un composant. Dans tous les cas, il est indispensable de disposer d'un moyen de décrypter l'information transmise avec précision, ce qui suppose l'existence d'une spécification standardisée [6] du composant en question.

2.3 Augmentation des volumes de données

La disponibilité de nombreux corpus de plus en plus volumineux, le succès des méthodes d'apprentissage qui en sont gourmandes et la concurrence de Google (voir par exemple (Michel *et al.*, 2011)), poussent la linguistique de corpus à continuellement augmenter les volumes de données traitées. Dans de nombreuses tâches, telles que la traduction automatique, la taille des ressources influe directement sur la performance de l'outil. Dans un tel contexte, la capacité à traiter de grandes quantités de données devient progressivement indispensable pour rester compétitif au niveau international. Or travailler sur de tels volumes de données nécessite la prise en compte de problèmes jusqu'ici délaissés par le TAL : traitement en parallèle, robustesse des échanges entre composants ou encore distribution des données entre chaînes exécutées en parallèles, etc. De telles problématiques ne sont pas nouvelles, elles ont été l'objet de recherches dans d'autres domaines et les techniques qui en sont issues sont communément utilisées dans l'industrie. Elles requièrent néanmoins une rigueur dans le développement, une homogénéisation des mécanismes d'interfaces entre composants, et une résistance accrue aux erreurs.

Nous pensons que sans un profond changement de perspective et de méthodologie dans la conception et la réalisation de composants, ces différents défis technologiques (qualité des composants, interopérabilité, masses de données) risquent de maintenir la communauté du TAL prisonnière de l'ingénierie logicielle, au détriment de la recherche. C'est pourquoi il nous est nécessaire de nous accorder sur des normes d'interopérabilité entre composants, de façon à pouvoir déléguer l'ingénierie des parties critiques de nos chaînes de traitement qui ne relèvent pas des traitements TAL à des spécialistes de l'ingénierie logicielle. Répondre à ce besoin est précisément l'objectif d'UIMA et Apache UIMA.

6. Ce type de processus de standardisation n'est en rien original, puisque des processus similaires ont permis un grand nombre d'avancées technologiques en informatique. Cependant dans le cas du TAL une telle standardisation est tout simplement impossible, car bien entendu les langues naturelles ne se laissent pas formaliser/catégoriser aussi facilement : la variété et l'évolutivité des langues rendent inévitables l'existence d'une multiplicité de représentations à tous les niveaux. C'est l'une des raisons pour lesquelles l'interopérabilité en TAL n'est pas seulement un défi technologique mais aussi scientifique : pour caricaturer, on pourrait dire que le standard XML est une réponse technique, mais on voit bien que ce format n'aide en rien à définir le *contenu* des données échangées. La question de standardiser la forme (pas seulement le format) sans contraindre le fond n'est pas triviale : voir la réponse proposée par UIMA en section 3.

3 L'approche UIMA

Le projet UIMA (*Unstructured Information Management Architecture*) est avant tout une spécification, validée par l'OASIS [7], garantissant l'interopérabilité de briques logicielles destinées à la structuration de données de différentes modalités [8] (texte, audio, vidéo...). Cette spécification ne vise pas à offrir une « boîte à outils » pratique ou des composants d'annotations clé en main, mais définit un *cadre* de développement de chaînes d'analyse de données non structurées.

Apache UIMA est une implémentation de ce standard initiée par IBM et désormais portée par la fondation Apache [9]. Plus exactement Apache propose deux implémentations distinctes : l'une en Java et la seconde en C++, cette dernière étant elle-même accessible à travers d'autres langages (Perl, Python, TCL) via un mécanisme d'interface.

La spécification UIMA encourage la réutilisation des composants de traitement afin de limiter la duplication des développements. Cet objectif est réalisé grâce à la définition d'une structure de données normalisée, le *CAS* (*Common Analysis Structure*), qui permet aux composants de communiquer le résultat de leurs traitements et garantit ainsi leur intéropérabilité, et par un système d'ordonnancement des composants en chaînes de traitement qui peuvent être exécutées en parallèle.

3.1 Interopérabilité des composants

Les composants d'analyse, appelés *analysis engines* (AE), sont des briques logicielles pilotables par une API particulière. Chaque composant est décrit par un fichier XML qui regroupe des informations d'identification (un nom unique, une description et un numéro de version), de configuration (noms des paramètres du composant et valeurs par défaut) et de comportement (dépendances en entrée et opérations fournies en sortie) qui permet à toute implémentation d'UIMA d'exécuter le composant indépendamment de la façon dont il est implémenté. Cet accès au composant par un fichier texte offre un premier niveau d'interopérabilité complété par le couple *CAS* (*Common Analysis Structure*) et *Type System* (TS).

Dans une chaîne de traitement les composants communiquent les uns avec les autres en s'échangeant des *CAS*. Ces derniers contiennent à la fois l'artefact à analyser [10] ainsi que les informations de structuration issues des analyses antérieures : les annotations. Lors de son exécution un composant reçoit en entrée un *CAS* contenant un ensemble d'annotations, qu'il peut exploiter (en partie ou en totalité) pour en produire de nouvelles ; il renvoit le *CAS* ainsi mis à jour avant d'analyser le *CAS* suivant.

Les annotations sont organisées au sein du *Type System* (TS) qui est partagé par tous les composants d'une même chaîne de traitement. Un TS est un ensemble de types (dont les annotations sont des instances) définis par le concepteur du composant pour répondre aux besoins de la tâche à accomplir. Un type est constitué d'un nom et d'un ensemble de traits, qui sont eux même des paires formées d'un nom et d'un type. UIMA introduit des types primitifs sans trait (Integer, String, Array...) ainsi qu'une algèbre de composition [11] de manière à pouvoir décrire tout type d'annotation.

Un tel modèle de représentation de la *structure logique des schémas d'annotations* (Bird & Liberman, 2001; Ide & Suderman, 2007) est comparable en termes d'objectifs avec d'autres travaux visant explicitement à favoriser l'interopérabilité tel que Ide & Suderman (2009). En pratique un même TS (ou un sous-ensemble du TS) est partagé par les différents composants susceptibles de travailler sur les annotations de ce(s) type(s) ; c'est pourquoi la conception du TS est un point critique dans la mise au point d'un ensemble de composants interopérables. Il est important de noter que le modèle proposé par UIMA est très flexible, de façon à permettre la représentation de tout type de méta-données tout en maintenant une représentation globale standardisée.

Ces caractéristiques permettent à un composant UIMA d'être inclus sans modification interne dans un programme,

7. L'OASIS est un consortium qui conduit le développement et l'adoption de normes ouvertes dédiées à la société de l'information. La norme UIMA v1.0 est disponible sur le site de l'OASIS : `http://docs.oasis-open.org/uima/v1.0/uima-v1.0.html`
8. Le texte reste clairement la modalité qui profite le mieux de la plateforme UIMA. Cependant, le projet SAPIR (Kaplan *et al.*, 2009) analyse des flux vidéos qu'il redécoupe en images et sons également au sein d'UIMA.
9. `http://uima.apache.org`
10. On nomme cet artefact le *SOFA* (*Subject of Analysis*)
11. Les types primitifs ainsi que l'algèbre de composition sont tirés du méta-modèle Ecore.

d'être déployé comme *web service* ou encore d'être alimenté depuis une grande variété de sources (fichier, sortie d'un autre composant, composant aspirateur de pages web, etc.). Un tel composant est potentiellement utilisable dans des contextes pour lesquels il n'a pas été conçu au départ, grâce au fait qu'il respecte une spécification précise.

3.2 Parallélisation des chaînes de traitements

L'exécution d'un traitement sur des données est décrite par un CPE (*Collection Processing Engine*). Un CPE est initié par un composant particulier, le *Collection Reader*, dont le rôle est de charger les artefacts dans le système sous forme de CAS qui sont ensuite distribués aux chaînes de traitements. Les CPE sont encore une fois décrits par un fichier XML de sorte que toute implémentation de la norme UIMA puisse l'exécuter.

Le *collection reader* est exécuté au sein du processus principal du CPE et ne peut-être parallelisé. Il s'agit donc, lorsque les analyses à mener sont légères, du principal goulot d'étranglement des données. Il est alors avantageux lorsque c'est possible de découper la collection d'artefacts à traiter en amont, à l'aide d'un système tel qu'Hadoop [12].

Une chaîne de traitement, appelée *processing pipeline*, est une collection de composants d'analyse (les *AE*) associée à un composant de routages des *CAS* (le *flow controler*). Ces chaînes de traitement sont autonomes ce qui permet leur duplication dans des processus distincts à des fins de parallélisation. UIMA intègre ainsi un système d'ordonnancement qui permet de distribuer les CAS produits par le *collection reader* entre les différentes instances des chaînes de traitement selon la charge de chacune. De plus, ces chaînes de traitement prennent en charge plusieurs problématiques classiques. D'abord, elles disposent d'un système robuste de gestion des erreurs qui permet d'éviter le plantage de la chaîne complète lorsqu'un composant rencontre une erreur avec un artefact particulier. Ensuite, elles permettent à plusieurs composants (ou plusieurs instances d'un même composant) de partager les mêmes ressources (dictionnaire, base de données, ...) ce qui allège considérablement la charge machine lorsque ces ressources sont volumineuses (réseaux sémantiques...).

3.3 Développement collaboratif communautaire

Les expérimentations en TAL reposent sur un mille-feuilles applicatif : outils de chargement des différents corpus ; découpage des textes, des sons ou des images en unités mots, phonèmes, graphèmes ; analyses morphologiques, syntaxiques... ; application de modèles de langage ; etc. Ces différentes briques logicielles, lorsqu'elles sont disponibles, collaborent difficilement les unes avec les autres. Les chercheurs en traitement automatique des langues doivent donc réussir à mettre en place une chaîne de prétraitement à partir de laquelle ils pourront envisager mettre en œuvre leurs expérimentations. À ce premier obstacle fastidieux s'ajoute souvent un protectionnisme injustifié des équipes envers leurs propres développements. Ces obstacles sont un frein à l'utilisation des outils les plus performants, donc à la qualité des expérimentations et indirectement à la recherche en général.

L'architecture à composants choisie pour UIMA ouvre la voie à un partage facilité des développements entre les différents acteurs de la recherche en TAL. Nous pensons qu'en encourageant une mutualisation des efforts par l'échange de briques logicielles performantes, nous ne pouvons que faciliter les travaux des membres de la communauté. Cette collaboration, si elle est désormais techniquement possible, nécessite de se concrétiser humainement. La communauté UIMA-Fr [13] a été récemment lancée dans ce but (Hernandez *et al.*, 2010).

4 Ouverture

Une introduction objective à UIMA ne doit pas omettre de mentionner l'inconvénient majeur à son utilisation : un coût élevé en temps. Tout d'abord le développeur doit acquérir les principes de bases d'UIMA (dont la courbe d'apprentissage est particulièrement raide), et même ensuite, programmer et utiliser des composants UIMA passent souvent par divers obstacles techniques, dont une grande partie n'existeraient pas dans le cas d'un composant

12. `http://hadoop.apache.org/`
13. La partie la plus visible de cette communauté reste le portail `uima-fr.org`

individuel réalisé sans contrainte. Actuellement le développement de composants UIMA est un investissement à moyen terme, dans l'objectif que les composants créés pourront interagir avec des composants externes de façon fructueuse. Mais pour toutes les raisons évoquées plus haut (section 2) cet investissement est aujourd'hui nécessaire pour la communauté : on peut comparer cette transition à celle qui s'est opérée depuis les langages de programmation non structurés aux langages évolués que nous connaissons aujourd'hui.

Il est intéressant de noter qu'à la suite d'une telle évolution la communauté du TAL (du moins sa partie académique) serait en mesure de mettre en commun ses moyens logiciels et humains de façon similaire à la communauté du développement *open source* : grâce à la mise au points d'outils et de mécanismes de dévelopement collaboratif standardisés, cette communauté est aujourd'hui à l'origine d'un grand nombre de produits logiciels dont la qualité est établie, ce qui ne peut que nous inspirer.

La question de la place de l'ingénierie dans le développement de nos outils ne doit pas être écartée sous prétexte de la confusion entre recherche et ingénierie en informatique. Le TAL est à la croisée de plusieurs autres domaines (la linguistique, les statistiques, le traitement du signal, travaux intermodalités...), et une part importante des travaux y consiste à adapter les résultats et les techniques issus de ces domaines. Cette adaptation comporte une part d'ingénierie spécialisée, qui est partie intégrante du processus de recherche : les données en langue naturelle présentent en effet des spécificités qui nécessitent un traitement adapté. Cette adaptation s'inscrit donc sans contradiction dans la définition de l'informatique comme science du traitement de l'information.

Références

BIRD S. & LIBERMAN M. (2001). A formal framework for linguistic annotation. *Speech Commun.*, **33**, 23–60.

HERNANDEZ N., POULARD F., VERNIER M. & ROCHETEAU J. (2010). Building a French-speaking community around UIMA, gathering research, education and industrial partners, mainly in Natural Language Processing and Speech Recognizing domains. In *Workshop Abstracts LREC 2010 Workshop 'New Challenges for NLP Frameworks'*, p. p64, La Valleta Malte.

IDE N. & SUDERMAN K. (2007). Graf : a graph-based format for linguistic annotations. In *Proceedings of the Linguistic Annotation Workshop*, LAW '07, p. 1–8, Stroudsburg, PA, USA : Association for Computational Linguistics.

IDE N. & SUDERMAN K. (2009). Bridging the gaps : interoperability for graf, gate, and uima. In *Proceedings of the Third Linguistic Annotation Workshop*, ACL-IJCNLP '09, p. 27–34, Stroudsburg, PA, USA : Association for Computational Linguistics.

KAPLAN A., MAMOU J., GALLO F. & SZNAJDER B. (2009). Multimedia feature extraction in the SAPIR project. In *Proceedings of the Biennial GSCL Conference 2009*.

MICHEL J.-B., SHEN Y. K., AIDEN A. P., VERES A., GRAY M. K., TEAM T. G. B., PICKETT J. P., HOIBERG D., CLANCY D., NORVIG P., ORWANT J., PINKER S., NOWAK M. A. & AIDEN E. L. (2011). Quantitative Analysis of Culture Using Millions of Digitized Books. *Science*, **331**(6014), 176–182.