AppFM, une plate-forme de gestion de modules de TAL

Paul Bui-Quang[1] Brigitte Grau[1,2] Patrick Paroubek[1]
(1) LIMSI, Campus universitaire bât 508 Rue John von Neumann, 91405 Orsay, France
(2) ENSIIE, 1 square de la résistance 91025 Evry
paul.bui-quang@limsi.fr, brigitte.grau@limsi.fr,
patrick.paroubek@limsi.fr

RÉSUMÉ

AppFM [1] est un outil à mi-chemin entre un environnement de création de chaînes modulaires de TAL et un gestionnaire de services systèmes. Il permet l'intégration d'applications ayant des dépendances complexes en des chaînes de traitements réutilisables facilement par le biais de multiples interfaces.

ABSTRACT

AppFM, a tool for managing NLP modules

AppFM is a tool between a NLP pipeline framework and a system service management. It allows integration of applications with complex dependencies into functional modules workflows of convenient usage within multiples interfaces.

MOTS-CLÉS : intégration, chaîne d'outils, orchestration de traitements, plate-forme TAL.

KEYWORDS: integration, pipeline management, process management, NLP workbench.

Dans le domaine du traitement automatique des langues les applications prennent souvent la forme de chaînes de processus qui s'appliquent en cascade sur un corpus et produisent des résultats intermédiaires. Ces unités de traitement spécialisées (normalisation, tokenization, analyse syntaxique, etc.) sont utilisées de manière récurrentes pour chaque projet applicatif à l'exception des parties sur lesquelles se focalise le travail de recherche. Il existe déjà des solutions permettant la mise en place de telles chaînes de traitement, par exemple les environnements UIMA (uim, 2013), GATE (gat, 2016) ou LingPipe (lin, 2011). Ces outils, une fois installés, permettent de créer relativement facilement des séquences de traitement. Ils se spécialisent ensuite dans des optiques industrielles ou d'expérimentation. Ces solutions sont néanmoins complexes à prendre en main dans un premier temps et peuvent imposer un modèle contraignant de développement. S'appuyant sur les nouvelles technologies de virtualisation et de programmation par message, AppFM (Application Frame Manager) est un outil qui opère sur un concept de chaîne plus abstrait et plus ouvert. Il tend à fournir des fonctionnalités s'approchant de la plateforme très aboutie LAPPS (Ide *et al.*, 2016).

La fonction principale de la plate-forme AppFM est d'orchestrer des processus avec des outils récents (docker, zmq) à mi-chemin entre des plate-formes comme Juju (juj, 2015) et des environnements tels UIMA. L'objectif est de permettre un déploiement facile de modules TAL sous forme de chaîne de traitement typique des applications de ce domaine et ce avec le plus de liberté possible pour le développeur de modules. Les 3 axes qui ont initialement motivé la réalisation d'une nouvelle plate-forme de traitement sont : a) La possibilité d'intégrer tout type d'application indépendamment

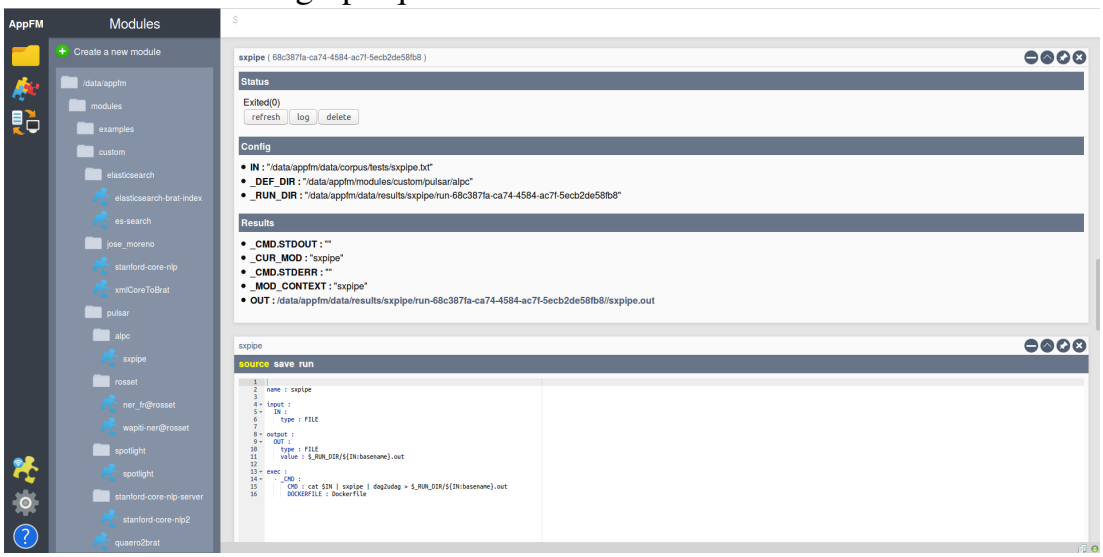
1. Ce travail a été financé dans le cadre des projets FUI 13 Projestimate et FUI 18 Pulsar

du langage de programmation et de la complexité des dépendances ; b) La prise en charge de la parallélisation des traitements ; c) La facilité d'utilisation.

Etant donnée la diversité toujours croissante des langages et des librairies, et malgré l'existence de modèles de programmations génériques (framework UIMA, ESB, etc.) le parti a été pris d'utiliser la technologie Docker pour virtualiser l'environnement propre à toute application ce qui permet, de plus, une plus grande fiabilité dans le déploiement et l'exécution des applications.

Nous y avons ajouté la formalisation des applications, i.e. des modules de TAL, sous forme fonctionnelle. Une unité de traitement est représentée comme une fonction ayant un espace de définition. Les entrées et sorties sont explicitement définies selon leurs types basiques (fichier, sortie standard, dossier, etc.) et leur arité. Cette approche, inspirée de la programmation par contrat, permet d'étendre ces types en précisant les formats et schémas de ces entrées et sorties. Cette définition, au format YAML, est complétée par l'explicitation du processus de traitement via une liste de commandes constituée d'opérateurs exécutés séquentiellement. Ces opérateurs permettent de lancer une commande shell (_CMD), de paralléliser une sous séquence de commandes sur une liste d'entrées (_MAP), d'exécuter des sous séquences conditionnées par la valeur d'une variable (_IF) ou d'exécuter un autre module. Ainsi, un module peut être vu à la fois comme une chaîne de traitement ou comme une unité de traitement réutilisable dans un autre module. De plus, des variables globales correspondant au module et à l'exécution en cours permettent une définition complète du processus représenté par un module.

FIGURE 1 – Interface graphique montrant un module et un résultat d'exécution.



Bien que développée dans le contexte du TAL, AppFM est fondamentalement agnostique concernant la sémantique des modules qui composent les chaînes de traitement qu'il orchestre. Au delà des modules qui définissent un traitement ponctuel, AppFM intègre également la notion de service et permet le déploiement d'outils telles que les bases de données ou des applications web. AppFM étant avant tout un serveur dont la principale fonction est d'exécuter des modules (de manière synchrone ou asynchrone), il est doté de deux types d'interfaces client facilitant différents types d'usages :

— Le client web permet de visualiser et gérer les modules graphiquement, est adapté pour des usages de démonstration, d'expérimentation rapide et est indiqué pour non experts en informatique.

— Le client en ligne de commande comprend toutes les fonctionnalités basiques de gestion des modules (lancement, status, etc.) et permet aux utilisateurs plus avancés de "scripter" avec les commandes fournies.

Références

(2011). Lingpipe - a tool kit for processing text using computational linguistics. `http://alias-i.com/lingpipe/`.

(2013). Apache uima - unstructured information management applications. `https://uima.apache.org/`.

(2015). Juju - an open source service orchestration management tool. `http://www.ubuntu.com/cloud/juju`.

(2016). Gate - general architecture for text engineering. `https://gate.ac.uk/`.

IDE N., SUDERMAN K., PUSTEJOVSKY J., VERHAGEN M. & CIERI C. (2016). The language application grid and galaxy. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.