Der TextImager als Front- und Backend für das verteilte NLP von Big Digital Humanities Data

Preprint

Wahed Hemati
Goethe-Universität Frankfurt
hemati@em.uni-frankfurt.de

Alexander Mehler
Goethe-Universität Frankfurt
mehler@em.uni-frankfurt.de

Giuseppe Abrami
Goethe-Universität Frankfurt
abrami@em.uni-frankfurt.de

Daniel Baumartz
Goethe-Universität Frankfurt
baumartz@stud.uni-frankfurt.de

April 5, 2019

***K*eywords** Natural Language Processing · NLP · Big Data · UIMA · Cluster Computing

Abstract

Immer mehr Disziplinen benötigen *Natural Language Processing* (NLP) Werkzeuge, um automatische Textanalysen auf verschiedenen Ebenen der Sprache durchzuführen. Die Anzahl der NLP-Werkzeuge wächst rasant[1]. Auch die Anzahl der frei oder anderweitig zugänglichen Ressourcen wächst. Angesichts dieser wachsenden Zahl an Werkzeugen und Ressourcen ist es schwierig, den Überblick zu behalten; gleichzeitig ist ein *Computational-Linguistic*-Framework, das große Datenmengen aus verschiedenen Quellen verarbeiten kann, noch nicht etabliert. Ein solches Framework sollte in der Lage sein, Daten verteilt zu verarbeiten und gleichzeitig eine standardisierte Programmier- und Modellschnittstelle bereitzustellen. Darüber hinaus sollte es modular und leicht erweiterbar sein, um die ständig wachsende Palette neuer Ressourcen und Tools zu integrieren. Das Framework muss offen genug für Erweiterungen Dritter sein, wobei jede Erweiterung für die gesamte Community zugänglich bleibt. Das Framework sollte es zudem Dritten ermöglichen, den Zugang zu ihren Erweiterungen zu beschränken, wenn dies beispielsweise durch Urheberrecht, geistiges Eigentum oder Datenschutz erforderlich ist. Um diesen Anforderungen gerecht zu werden, haben wir den TextImager [4] um ein verteiltes Serversystem mit Cluster-Computing-Funktionen auf der Basis von UIMA [2] weiterentwickelt.

UIMA ist ein Framework zur Verwaltung von Datenflüssen zwischen Komponenten. Es bietet standardisierte Interfaces zur Erstellung von Komponenten an. Dabei können die Komponenten einzeln oder im Verbund in einer Pipeline-Struktur ausgeführt werden. UIMA bietet weitgehende Möglichkeiten der sequenziellen Ordnung von NLP-Werkzeugen und verspricht, auch in Zukunft von der Community weiterentwickelt zu werden: Prozess-Management auf der Basis von UIMA erscheint nach derzeitigem Stand daher als erste Wahl im Bereich von NLP und DH.

TextImager bietet eine Vielzahl von UIMA-basierten NLP-Komponenten an, darunter unter anderen einen *Tokenizierer*, einen *Lemmatisierer*, einen *Part-Of-Speech-Tagger*, einen *Named-Entity-Parser* und einen *Dependency Parser*, und zwar für eine Vielzahl von Sprachen, darunter Deutsch, Englisch, Französisch und Spanisch. Dieses Spektrum an Werkzeugen besteht allerdings nicht ausschließlich aus Eigenentwicklungen, sondern wird maßgeblich um Entwicklungen Dritter erweitert, wozu unter anderem die Tool-Palette von *Stanford CoreNLP* [5], *OpenNLP* [6] und *DKPro* [1] zählen.

In Zeiten von *Big Data* wird es immer relevanter, Daten schnell zu verarbeiten. Aus diesem Grund ist TextImager als Multi-Server- und zugleich als Multi-Instanz-Cluster aufgebaut, um das verteilte Verarbeiten von Daten zu

[1] https://github.com/topics/nlp

ermöglichen. Dafür setzt TextImager auf UIMAs Cluster-Management-Dienste UIMA-AS[2] und UIMA-DUCC[3] auf.

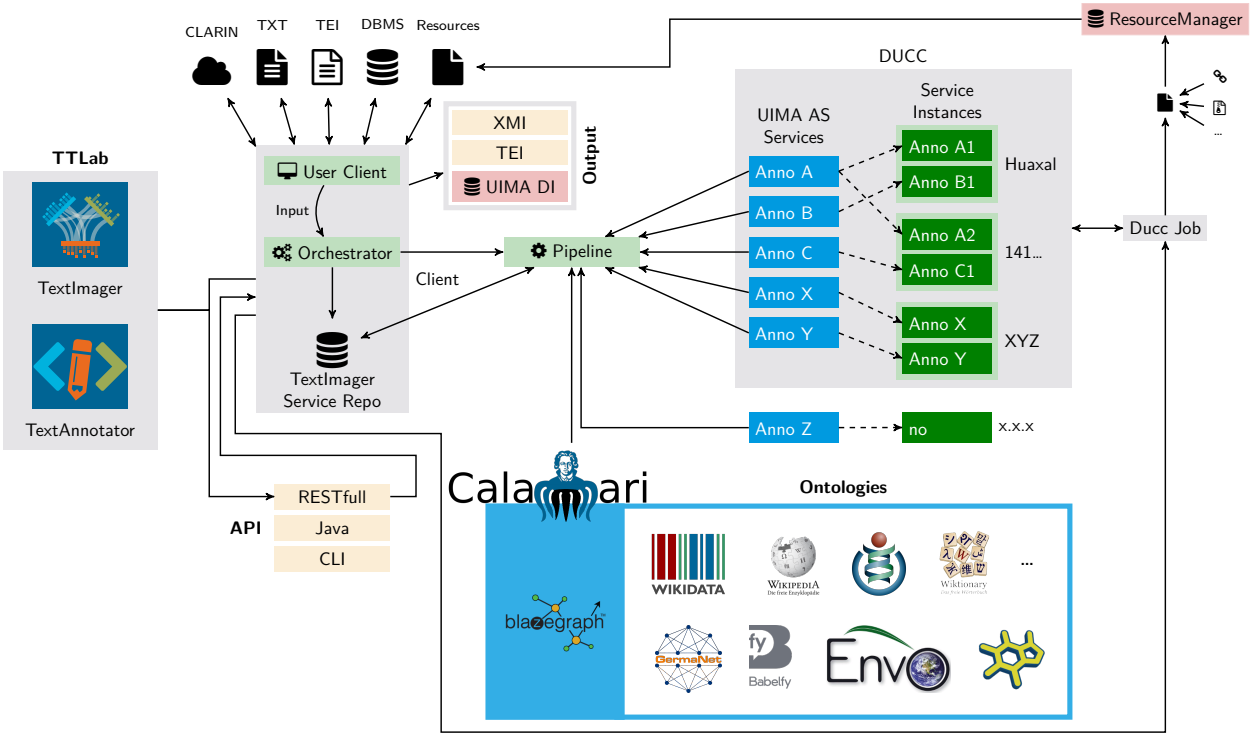

Figure 1: TextImager: system components and their relations.

Abbildung 1 zeigt eine schematische Darstellung von TextImager. Jede NLP-Komponente läuft als UIMA-AS Webservice auf dem Computing-Cluster des TextImager. Dabei können mehrere Instanzen einer Komponente instanziiert (s. Abbildung 1, *Service Instances*) werden und dennoch über eine Webservice-Schnittstelle (s. Abbildung 1, *UIMA AS Services*) angesprochen werden. Dazu wird das Java Messaging Service (JMS) verwendet, das die Kommunikation zwischen verschiedenen Komponenten einer verteilten Anwendung ermöglicht. JMS implementiert ein Point-to-Point-Kommunikationssystem. Dieser Kommunikationstyp basiert auf dem Konzept der *message queues* (Warteschlangen), *senders* (Sender) und *receivers* (Empfänger). Jedem Dienst ist eine Eingabewarteschlange und eine Ausgabewarteschlange zugeordnet. Um mehrere Instanzen einer Komponente zu verteilen, verbinden sich die Instanzen mit der gleichen Service-Eingangswarteschlange. Die Instanzen erhalten aus dieser Warteschlange Arbeitseinheiten. Nach der Verarbeitung wird das Ergebnis an eine Ausgabewarteschlange zurückgegeben. Die Ausgabewarteschlange eines Dienstes kann an eine Eingabewarteschlange eines anderen Dienstes angeschlossen werden, um eine Pipeline zu erstellen. Aufgrund dieser Ein- und Ausgabewarteschlangen-Systematik kann jeder Service Arbeitseinheiten asynchron bearbeiten. Durch diese Architektur ist TextImager eine Multi-Server-, Multi-Service- und Multi-Service-Instanz-Architektur.

Darüber hinaus bietet TextImager ein Toolkit, das es jedem Entwickler ermöglicht, einen eigenen TextImager-Cluster aufzusetzen und Services im TextImager-System hinzuzufügen. Entwickler können den Zugriff auf die Dienste einschränken, wenn dies wie oben beschrieben erforderlich ist, was mittels die Integration des *ResourceManagers* [3] und des *AuthorityManagers* [3] realisiert wird.

Durch Freigabe des Quellcodes des TextImager und die Bereitstellung von Leitlinien für dessen Erweiterung wollen wir es Dritten ermöglichen, ihre NLP-Software über die Webservices von TextImager zu vertreiben, so dass die gesamte wissenschaftliche Gemeinschaft davon profitiert. Installationsanweisungen und Beispiele für die Einrichtung eines TextImager-Servers finden Nutzer in folgendem GitHub-Repository: `https://github.com/texttechnologylab/textimager-server`.

[2] `https://uima.apache.org/doc-uimaas-what.html`
[3] `https://uima.apache.org/doc-uimaducc-whatitam.html`

Der Beitrag erörtert die Möglichkeiten und Grenzen des NLP von Big Data, stellt den TextImager als Werkzeug für diesen Bereich zur Diskussion und zeigt anhand von drei Nutzungsszenarien Einsatzmöglichkeiten in den DH auf.

References

[1] Richard Eckart de Castilho and Iryna Gurevych. A broad-coverage collection of portable NLP components for building shareable analysis pipelines. In *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT*, pages 1–11, Dublin, Ireland, August 2014. Association for Computational Linguistics and Dublin City University.

[2] David Ferrucci and Adam Lally. UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3-4):327–348, 2004.

[3] Rüdiger Gleim, Alexander Mehler, and Alexandra Ernst. SOA implementation of the ehumanities desktop. In *Proceedings of the Workshop on Service-oriented Architectures (SOAs) for the Humanities: Solutions and Impacts, Digital Humanities 2012, Hamburg, Germany*, 2012.

[4] Wahed Hemati, Tolga Uslu, and Alexander Mehler. Textimager: a distributed uima-based system for nlp. In *Proceedings of the COLING 2016 System Demonstrations*. Federated Conference on Computer Science and Information Systems, 2016.

[5] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

[6] OpenNLP. Apache OpenNLP, `http://opennlp.apache.org`, 2010.