TextImager: a Distributed UIMA-based System for NLP

Wahed Hemati
Text Technology Lab
Goehte-Universität Frankfurt

Tolga Uslu
Text Technology Lab
Goehte-Universität Frankfurt

Alexander Mehler
Text Technology Lab
Goehte-Universität Frankfurt

{hemati,uslu,mehler}@em.uni-frankfurt.de
http://www.hucompute.org

Abstract

More and more disciplines require NLP tools for performing automatic text analyses on various levels of linguistic resolution. However, the usage of established NLP frameworks is often hampered for several reasons: in most cases, they require basic to sophisticated programming skills, interfere with interoperability due to using non-standard I/O-formats and often lack tools for visualizing computational results. This makes it difficult especially for humanities scholars to use such frameworks. In order to cope with these challenges, we present TextImager, a UIMA-based framework that offers a range of NLP and visualization tools by means of a user-friendly GUI. Using TextImager requires no programming skills.

1 Introduction

Computational humanities and related disciplines require a wide range of NLP tools to perform automatic text analyses on various levels of textual resolution. This includes, for example, humanities scholars dealing with repositories of historical documents, forensic linguists analyzing unstructured texts of online social media to create digital fingerprints of suspects or even doctors using clinical NLP to support differential diagnosis based on physician-patient talks. However, established NLP frameworks still require basic to sophisticated programming skills for performing such analyses. This hampers their usage for users who are not sufficiently trained neither in computational linguistics nor in computer science. Further, these frameworks often lack interoperability due to using non-standard I/O-formats. We present TextImager to cope with these challenges. The longer-term goal of TextImager is to provide a platform into which any open source/access NLP tool can be integrated. To this end, TextImager provides a web-based GUI whose usage does not require any programming skills while making accessible a range of tools for visualizing results of text analyses. In order to ensure standardization and interoperability, TextImager is based on the *Unstructured Information Management Applications* (UIMA) framework. Currently, the scope of TextImager ranges from tokenizing, lemmatizing, POS-tagging, text similarity measurements to sentiment analysis, text classification, topic modeling and many more.

2 Related Work

Frameworks of computational texts analysis have already been introduced and are now common in industrial use. This includes, for example, UIMA (Ferrucci and Lally, 2004), DKPro (Eckart de Castilho and Gurevych, 2014), OpenNLP (OpenNLP, 2010) and Gate (Cunningham et al., 2011). Note that these frameworks do not provide visualization interfaces and require versatile programming skills for set up. Thus, they cannot be recommended for being used by computationally less trained users. We provide the TextImager to cope with this problem while integrating most of the components of these frameworks. On the other hand, Voyant Tools (Bird et al., 2009; Ruecker et al., 2011), WebNLP (Burghardt et al., 2014) and conTEXT (Burghardt et al., 2014) are web-based NLP tools including visualization components. In order to combine the best of both worlds, TextImager additionally subsumes the functionalities

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations, pages 59–63, Osaka, Japan, December 11-17 2016.

of these tools. It also shares functionalities with WebLicht (Hinrichs et al., 2010). However, unlike WebLicht, TextImager is based on open UIMA and, thus, complies to an industrial standard of modeling text processing chains.

3 System Architecture of TextImager

TextImager consists of two parts, front-end and back-end. The front-end is a web application that makes all functionalities and NLP processes available in a user-friendly way. It allows users for analyzing and visualizing unstructured texts and text corpora. The back-end is a highly modular, expandable, scalable and flexible architecture with parallel processing capabilities.

3.1 Back-end

Figure 1 shows the architecture of TextImager. Every NLP component of TextImager implements a UIMA interface. Every UIMA compatible NLP-component can easily be integrated into TextImager. Even modules not compatible with UIMA can be integrated with just a slight effort. Amongst others, we have integrated DKPro (see Section 2), which offers a variety of UIMA-components. We also integrated the *UIMA Asynchronous Scaleout* (UIMA-AS)[1] add-on.

TextImager allows users for dynamically choosing NLP components in a pipeline. To this end, we extended UIMA-AS by initiating components without XML descriptors by means of uimaFIT[2]. We extended this framework by allowing for dynamic instantiations of pipelines. These extensions make our framework highly flexible, adaptive and extensible during runtime.

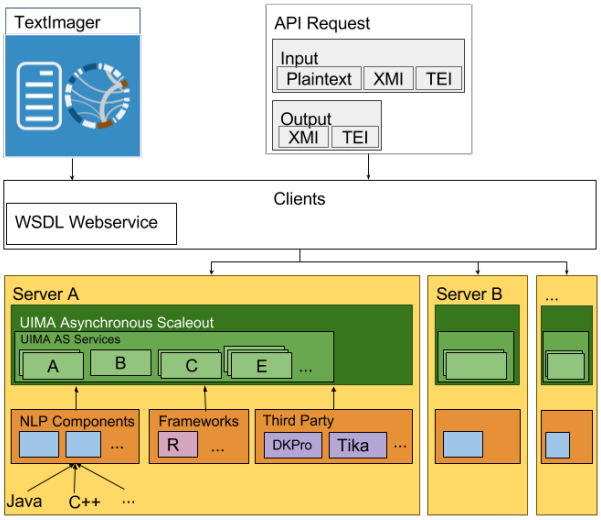
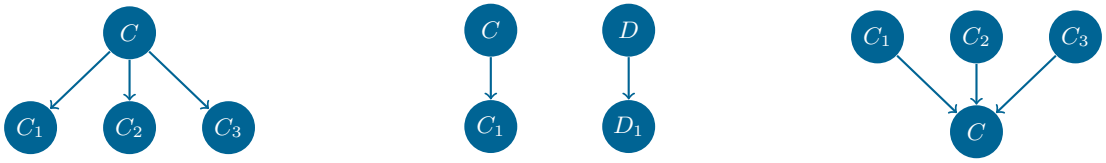


Figure 1: TextImager's back-end.

All TextImager components are configured as UIMA-AS services, which may run standalone or in a pipeline. All services are located on servers to allow for communication among them. Note that we are not limited to run these components on a single server; rather, they can be distributed among different servers (see Figure 1). We developed a mechanism that automatically selects and acquires components and their resources: it arranges components into pipelines and grants the ability to parallelize them. Thus, components that do not depend on each other can run in parallel. For this we developed an advanced UIMA flow controller. Take the examples displayed in Fig.2: suppose that vertices in these examples denote NLP components; suppose further that the corresponding arcs denote interdependencies between these components. In Fig. 2a, the components C_1, C_2 and C_3 do not depend on each other. Thus, they can run in parallel. In Fig. 2b, components C_1 and D_1 do not depend on each other, but on C and D, respectively. Thus, C and D can run in parallel as can do the components C_1 and D_1. In Fig. 2c, C depends on C_1, C_2 and C_3. Thus, running C has to wait on the termination of C_1, C_2 and C_3. Within TextImager, dependency hierarchies of components as exemplified by these three examples are generated from information provided by each of the components supposed that their input and output types have been defined appropriately (cf. the class specifications of type `org.apache.uima.fit.descriptor.TypeCapability`). In this way, TextImager allows for realizing a wide range of processing chains.

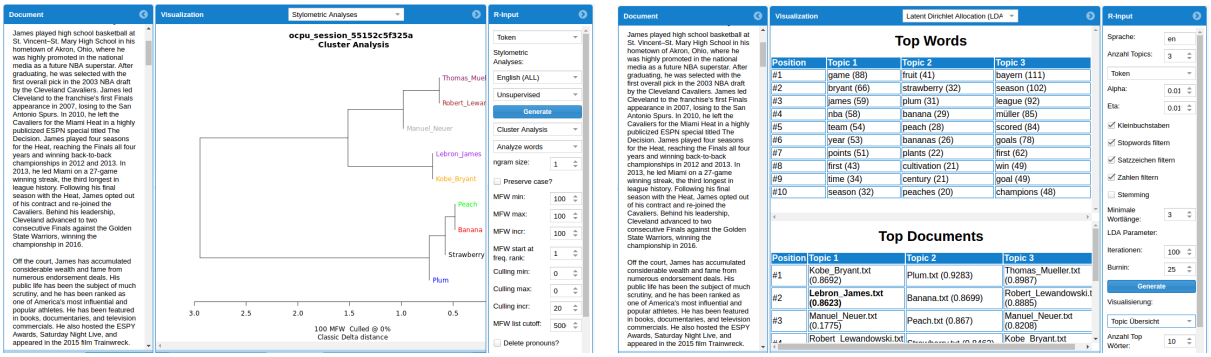
One advantage of our framework is that it does not rely on a central repository. Rather, TextImager can be distributed across multiple servers. This allows developers for setting up their own TextImager

[1] `https://uima.apache.org/doc-uimaas-what.html`
[2] `https://uima.apache.org/uimafit.html`



(a) (b) (c)

Figure 2: Component dependency types.



(a) Stylometric R-Module (b) LDA R-Module

server and to distribute their own NLP tools within the TextImager ecosystem.

TextImager can be used within a web application that offers a graphical user interface. Alternatively, TextImager can be used via a WSDL webservice client.

3.2 Front-end

The front-end gives access to all NLP tools integrated into TextImager without requiring any programming skills. This is done by means of a GUI that even provides three-dimensional text visualizations (see Figure 4b). All visualizations are interactive in the sense of allowing for focusing and contextualizing results of text analysis (e.g., the macro *reference distribution of sentence similarity across multiple documents* exemplified in Figure 4d). The GUI contains a text and a visualization panel. One of TextImager's guiding principles is to enable *bidirectional interactivity*. That is, any interaction with the visualization panel is synchronized by automatically adjusting the content of the text panel and vice versa. The front-end is based on *Ext JS*, a JavaScript framework for building interactive cross platform web applications. The visualizations are done by means of *D3.js*[3] and *vis.js*[4] to enable browser-based visualizations while handling large amounts of data.

Figure 4 exemplifies TextImager. With a focus on close reading, TextImager supports the interpretation of single texts by determining, for example, their central topics or by depicting their unfolding from constituent to constituent (see Figure 4g, 4a, 4h). Regarding distant reading (Jänicke et al., 2015), TextImager provides more abstract overviews of the content of text corpora. Here, visualizations provide summary information as exemplified in Figure 4b, 4c, 4d, 4f.

Last but not least, TextImager provides a generic interface to *R*[5]. The aim is to give access to any NLP-related package in R *once more without requiring programming skills*. This is especially needed for scholars in digital humanities who are not trained in using script languages for modeling statistical procedures, but expect a versatile tool encapsulating this computational complexity. Thus, TextImager users can process input texts using R packages like LDA (see Figure 3b), network analysis or stylometrics (see Figure 3a) without the need to manipulate or to invoke any *R* script directly. All these R packages

[3] https://www.d3js.org
[4] http://visjs.org
[5] https://www.r-project.org

are given a single entrance point in the form of TextImager. See (Mehler et al., 2016) for a recent research study based on TextImager.

4 Future Work

In already ongoing work, we extend the functionality of TextImager. This includes covering all features of tools like conTEXT. In contrast to many current frameworks, we will make TextImager's source code open-source as soon as the framework reaches a stable and documented version. We are going to specify a comprehensive model for component specification. The model will contain specifications of general components and their dependency hierarchy. This model will help defining where new NLP components are settled within the NLP landscape.

5 Scope of the Software Demonstration

A beta version of TextImager's web application can be found at `http://textimager.hucompute.org`. A preprocessed demonstration can be found at `http://textimager.hucompute.org/index.html?viewport=demo`. A tutorial on how to set up TextImager's backend services on codebase and a list of available components and options can be found at `http://service.hucompute.org`.

Acknowledgment

We gratefully acknowledge financial support of this project via the BMBF Project CEDIFOR (`https://www.cedifor.de/en/`).

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly, Beijing.

Manuel Burghardt, Julian Pörsch, Bianca Tirlea, and Christian Wolff. 2014. WebNLP – an integrated web-interface for Python NLTK and Voyant. In *KONVENS*, pages 235–240.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Niraj Aswani, Ian Roberts, Genevieve Gorrell, Adam Funk, Angus Roberts, Danica Damljanovic, Thomas Heitz, Mark A. Greenwood, Horacio Saggion, Johann Petrak, Yaoyong Li, and Wim Peters. 2011. *Text Processing with GATE (Version 6)*.

Richard Eckart de Castilho and Iryna Gurevych. 2014. 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. ACL and Dublin City University.

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

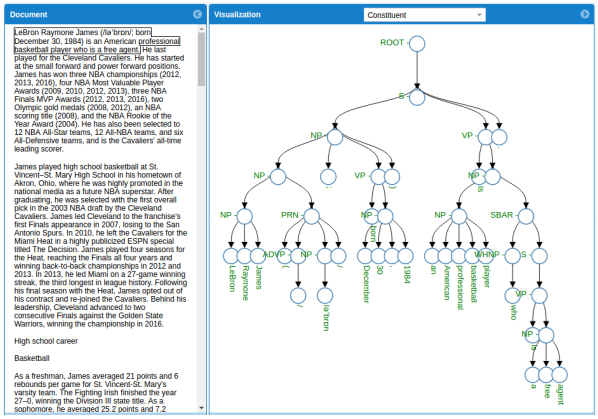
Erhard Hinrichs, Marie Hinrichs, and Thomas Zastrow. 2010. Weblicht: Web-based LRT services for German. In *Proceedings of the ACL 2010 System Demonstrations*, ACLDemos '10, pages 25–29, Stroudsburg, PA, USA. Association for Computational Linguistics.

Stefan Jänicke, Greta Franzini, M Cheema, and Gerik Scheuermann. 2015. On close and distant reading in digital humanities: A survey and future challenges. *Proc. of EuroVisSTARs*, pages 83–103.

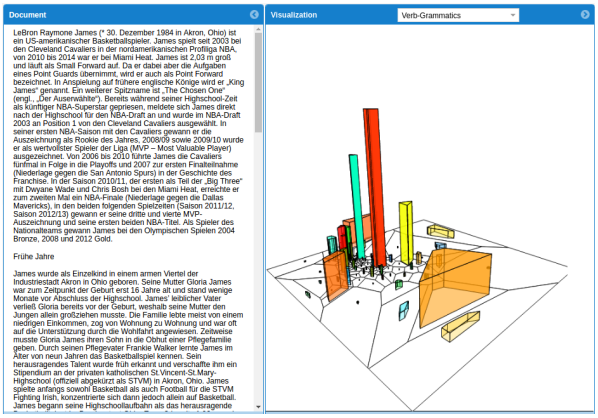
Alexander Mehler, Tolga Uslu, and Wahed Hemati. 2016. An image-driven approach to differential diagnosis. In *Proceedings of the 5th Workshop on Vision and Language (VL'16) hosted by the 54th Annual Meeting of the Association for Computational Linguistics (ACL), Berlin*, pages 80–85.

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

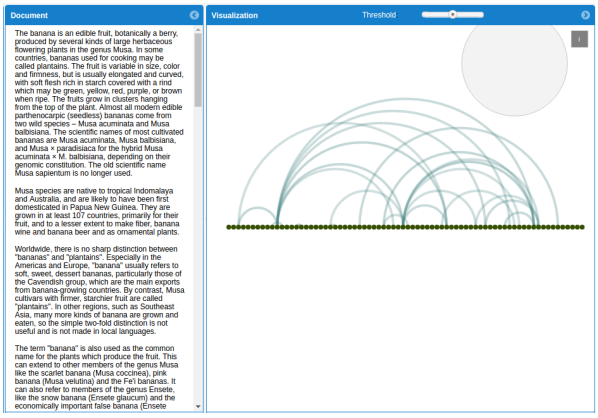
Stan Ruecker, Milena Radzikowska, and Stéfan Sinclair. 2011. *Visual interface design for digital cultural heritage: A guide to rich-prospect browsing*. Ashgate Publishing, Ltd.



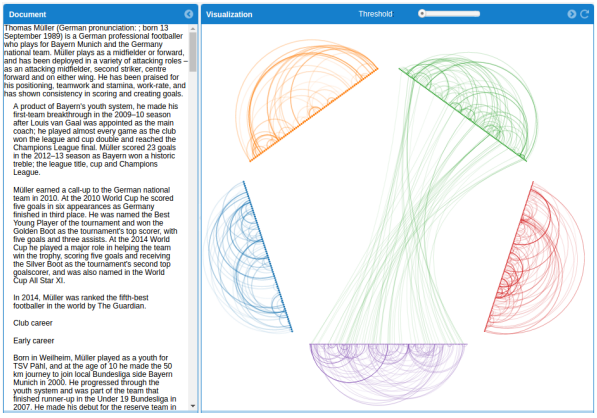
(a) Constituent parse tree



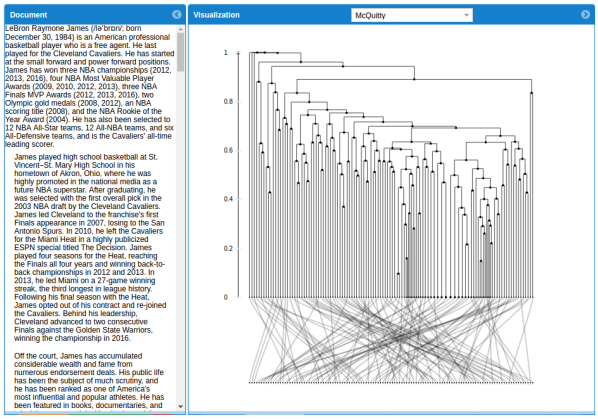
(b) Text2Voronoi



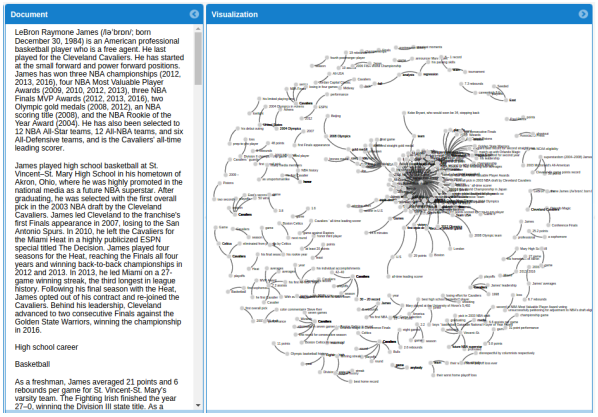
(c) Innertextual similarity



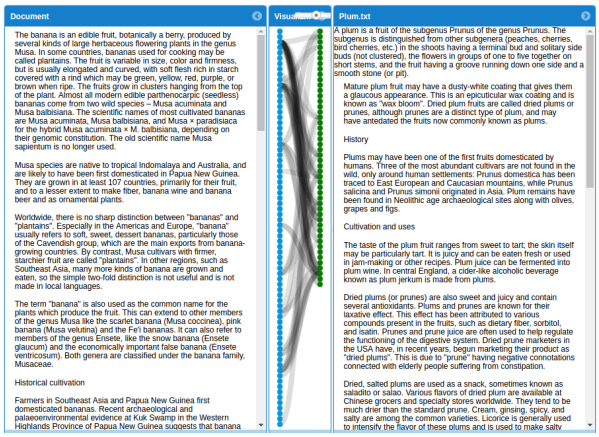
(d) Intertextual similarity



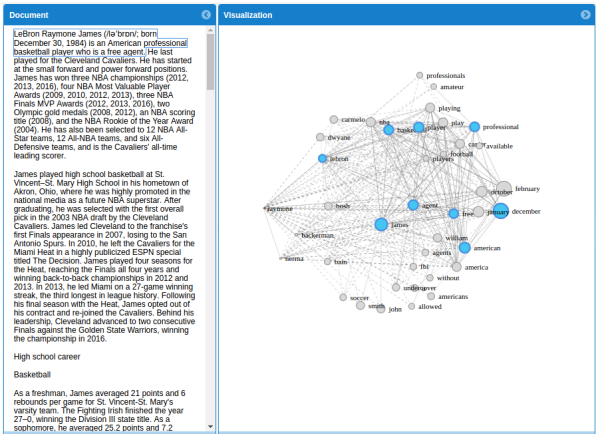
(e) Dendrogramcluster similarity



(f) Relation graph



(g) Bipartite similarity



(h) Semantic relation graph

Figure 4: Visualization Examples