TextImager as a Generic Interface to R

Tolga Uslu[1], Wahed Hemati[1], Alexander Mehler[1], and Daniel Baumartz[2]

[1,2]Goethe University of Frankfurt
[1,2]TextTechnology Lab
[1]{uslu,hemati,mehler}@em.uni-frankfurt.de
[2]baumartz@stud.uni-frankfurt.de

Abstract

R is a very powerful framework for statistical modeling. Thus, it is of high importance to integrate R with state-of-the-art tools in NLP. In this paper, we present the functionality and architecture of such an integration by means of *TextImager*. We use the *OpenCPU API* to integrate *R* based on our own *R*-Server. This allows for communicating with *R*-packages and combining them with *TextImager*'s NLP-components.

1 Introduction

We introduced *TextImager* in (Hemati et al., 2016) where we focused on its architecture and the functions of its backend. In this paper, we present the functionality and architecture of *R* interfaced by means of *TextImager*. For this purpose, we created a separate panel in *TextImager* for *R*-applications. In this panel, we combine state-of-the-art NLP tools embedded into *TextImager* with the the powerful statistics of *R* (R Development Core Team, 2008). We use the *OpenCPU API* (Ooms, 2014) to integrate *R* into *TextImager* by means of our own *R*-server. This allows for easily communicating with the built-in *R*-packages and combining the advantages of both worlds. In the case of topic detection (`LDA`), for example, the complete text is used as an input string to *R*. Thanks to *TextImager*'s preprocessor, more information is provided about syntactic words, parts of speech, lemmas, grammatical categories etc., which can improve topic detection. Further, the output of R routines is displayed by means of *TextImager*'s visualizations, all of which are linked to the corresponding input text(s). This allows for unprecedented *interaction* between text and the statistical results computed by *R*. For this paper we sampled several Wikipedia articles to present all features of *R* integrated into *TextImager*. This includes articles about four politicians (*Angela Merkel*, *Barack Obama*, *Recep (Tayyip) Erdoğan*, *Donald Trump*) and five sportsman, that is, three basketball players (*Michael Jordan*, *Kobe Bryant* and *Lebron James*) and two soccer players (*Thomas Müller* and *Bastian Schweinsteiger*).

2 Related Work

R is used and developed by a large community covering a wide range of statistical packages. The *CRAN*[1] package repository is the main repository of the *R* project. It currently consists of about 10 000 packages including packages for NLP.[2] The *R* framework requires basic to versatile skills in programming and scripting. It provides limited visualization and interaction functionalities. Attempts have been undertaken to provide web interfaces for *R* as, for example, *Shiny*[3] and *rApache*[4]. Though they provide a variety of functions and visualizations[5], these tools are not optimized for statistical NLP: their NLP-related functionalities are rather limited. In order to fill this gap, we introduce *TextImager*'s *R* package, that is, a web based tool for NLP utilizing *R*.

3 Architecture

TextImager is a *UIMA*-based (Ferrucci and Lally, 2004) framework that offers a wide range of NLP and visualization tools by means of a user-friendly *GUI* without requiring programming skills. It consists of two parts: front-end and back-end. The back-end is a modular, expandable, scalable and

[1]https://cran.r-project.org/
[2]https://cran.r-project.org/web/views/NaturalLanguageProcessing.html
[3]http://shiny.rstudio.com/
[4]http://rapache.net/
[5]http://shiny.rstudio.com/gallery/

flexible architecture with parallel and distributed processing capabilities (Hemati et al., 2016). The front-end is a web application that makes NLP processes available in a user-friendly way with responsive and interactive visualizations (Hemati et al., 2016). *TextImager* already integrated many third party tools. One of these is *R*. This section describes the technical integration and utilization of *R* into *TextImager*.

3.1 R / OpenCPU

R is a software environment for statistical computing. It compiles and runs on a variety of UNIX and Windows platforms. One of our goals is to provide an easy to use interface for R with a focus on NLP. To this end, we use *OpenCPU* to integrate *R* into *TextImager*. *OpenCPU* provides an *HTTP API*, which allocates the functionalities of R-packages (Ooms, 2014). The *OpenCPU* software can be used directly in *R*; alternatively, it can be installed on a server. We decided for the latter variant. In addition, we used the *opencpu.js* JavaScript library which simplifies API use in JavaScript and allows for calling *R*-functions directly from *TextImager*. To minimize the communication effort between client and server and to encapsulate *R* scripts from *TextImager*'s functionality, we created a so called *TextImager-R*-package that takes *TextImager* data, performs all *R*-based calculations and returns the results. This package serves for converting any *TextImager* data to meet the input requirements of any *R*-package. In this way, we can easily add new packages to *TextImager* without changing the HTTP request code. Because some data and models have a long build time we used *OpenCPU*'s session feature to keep this data on the server and access it in future sessions so that we do not have to recreate it. This allows the user for quickly executing several methods even in parallel without recalculating them each time.

3.2 Data Structure

The data structure of *TextImager* differs from *R*'s data structure. Therefore, we developed a generic mapping interface that translates the data structure from *TextImager* to an *R*-readable format. Depending on the *R*-package, we send the required data via *OpenCPU*. This allows for combining each NLP tool with any *R*-package.

3.3 OpenCPU Output Integration

Visualizing the results of a calculation or sensitivity analysis is an important task. That is why we provide interactive visualizations to make the information available to the user more comprehensible. This allows the user to interact with the text and the output of R, for example, by highlighting any focal sentence in a document by hovering over a word or sentence graph generated by means of R.

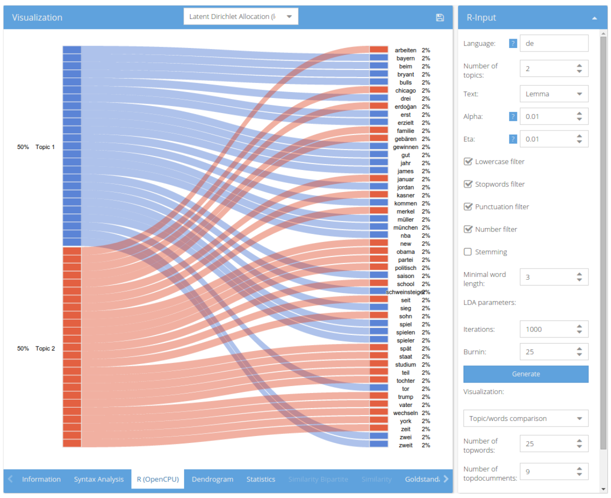
3.4 R-packages

This section gives an overview of *R*-packages embedded into the pipeline of *TextImager*.

tm The *tm*-package (Feinerer and Hornik, 2015) is a text mining *R*-package containing preprocess methods for data importing, corpus handling, stopword filtering and more.

lda The *lda*-package (Chang, 2015) provides an implementation of *Latent Dirichlet Allocation* algorithms. It is used to automatically identify topics in documents, to get top documents for these topics and to predict document labels. In addition to the tabular output we developed an interactive visualization, which assigns the decisive words to the appropriate topic (see Figure 1). This visualization makes it easy to differentiate between the two topics and see which words classify these topics. In our example, we have recognized words such as *players*, *season* and *game* as one topic (sportsman) and *party*, *state* and *politically* as a different topic (politics). The parameters of every package can be set on runtime. The utilization and combination of *TextImager* and *R* makes it possible, to calculate topics not only based on wordforms, but also takes other features into account like lemma, pos-tags, morphological features and more.

stylo The *stylo*-package (Eder et al., 2016) provides functionality for stylometric analyses. All parameters of the package can be set through the graphical user interface of TextImager. The package provides multiple unsupervised analyses, mostly based on a most-frequent-word list and contrastive text analysis. In Figure 2 we have calculated a cluster analysis based on our example corpus. We can see that the politicians, basketball players


Visualization | Latent Dirichlet Allocation (l

R-Input

Language: de

Number of topics: 2

Text: Lemma

Alpha: 0.01

Eta: 0.01

Lowercase filter

Stopwords filter

Punctuation filter

Number filter

Stemming

Minimal word length: 3

LDA parameters:

Iterations: 1000

Burnin: 25

Generate

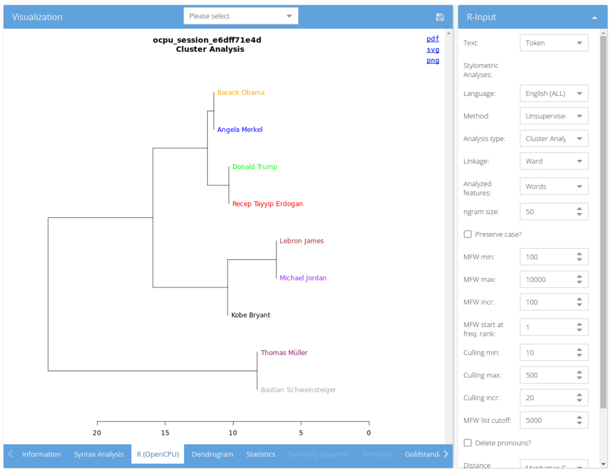
Visualization: Topic/words comparison

Number of topwords: 25

Number of topdocuments: 9

Figure 1: Words assigned to their detected topics

and soccer players have been clustered into their own groups.


Visualization | Please select

ocpu_session_e6dff71e4d
Cluster Analysis

R-Input

Text: Token

Stylometric Analyses:

Language: English (ALL)

Method: Unsupervise

Analysis type: Cluster Analy

Linkage: Ward

Analyzed features: Words

ngram size: 50

Preserve case?

MFW min: 100

MFW max: 10000

MFW incr: 100

MFW start at freq. rank: 1

Culling min: 10

Culling max: 500

Culling incr: 20

MFW list cutoff: 5000

Delete pronouns?

Distance: Manhattan

Figure 2: Cluster analysis of the documents

LSAfun The *LSAfun* (Günther et al., 2015) and *lsa* (Wild, 2015) packages provide functionality for *Latent Semantic Analysis*. In *TextImager* it is used to generate summaries of documents and similarities of sentences.

igraph The *igraph*-package (Csardi and Nepusz, 2006) provides multiple network analysis tools. We created a document network by linking each word with its five most similar words based on word embeddings. The *igraph*-package also allows to layout the graph and calculate different centrality measures. In the end, we can export the network in many formats (*GraphML*, *GEXF*, *JSON*, etc.) and edit it with graph editors. We also build an interactive network visualization (see figure 3) using *sigma.js* to make it interoperable with *TextImager*.

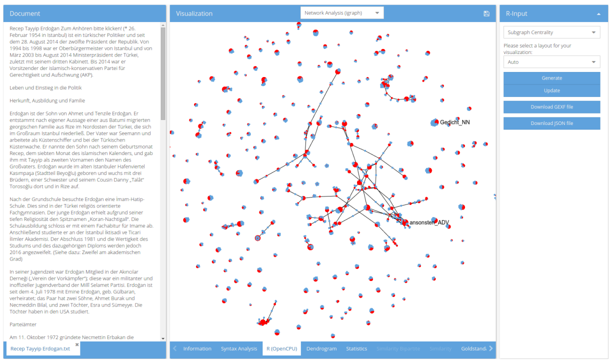
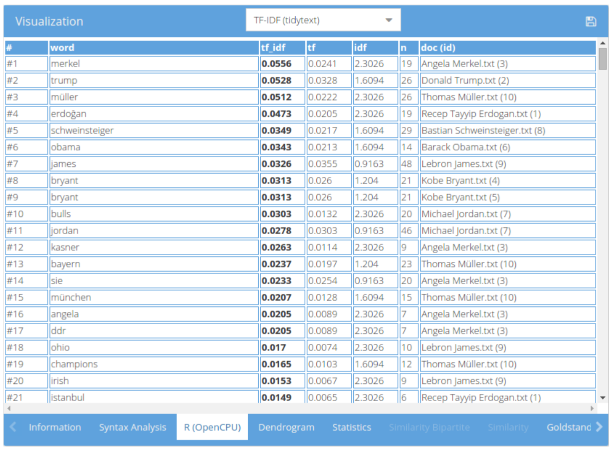


Figure 3: Network graph based on word embeddings

tidytext The *tidytext* package (Silge and Robinson, 2016) provides functionality to create datasets following the tidy data principles (Wickham, 2014). We used it with our *tm* based corpus to calculate *TF-IDF* information of documents. In Figure 4 we see the output tabular with informations like *tf*, *idf* and *tf-idf*


Visualization | TF-IDF (tidytext)

#	word	tf_idf	tf	idf	n	doc (id)
#1	merkel	**0.0556**	0.0241	2.3026	19	Angela Merkel.txt (3)
#2	trump	**0.0528**	0.0328	1.6094	26	Donald Trump.txt (2)
#3	müller	**0.0512**	0.0222	2.3026	26	Thomas Müller.txt (10)
#4	erdoğan	**0.0473**	0.0205	2.3026	19	Recep Tayyip Erdogan.txt (1)
#5	schweinsteiger	**0.0349**	0.0217	1.6094	29	Bastian Schweinsteiger.txt (8)
#6	obama	**0.0343**	0.0213	1.6094	14	Barack Obama.txt (6)
#7	james	**0.0326**	0.0355	0.9163	48	Lebron James.txt (9)
#8	bryant	**0.0313**	0.026	1.204	21	Kobe Bryant.txt (4)
#9	bryant	**0.0313**	0.026	1.204	21	Kobe Bryant.txt (5)
#10	bulls	**0.0303**	0.0132	2.3026	20	Michael Jordan.txt (7)
#11	jordan	**0.0278**	0.0303	0.9163	46	Michael Jordan.txt (7)
#12	kasner	**0.0263**	0.0114	2.3026	9	Angela Merkel.txt (3)
#13	bayern	**0.0237**	0.0197	1.204	23	Thomas Müller.txt (10)
#14	sie	**0.0233**	0.0254	0.9163	20	Angela Merkel.txt (3)
#15	münchen	**0.0207**	0.0128	1.6094	15	Thomas Müller.txt (10)
#16	angela	**0.0205**	0.0089	2.3026	7	Angela Merkel.txt (3)
#17	ddr	**0.0205**	0.0089	2.3026	7	Angela Merkel.txt (3)
#18	ohio	**0.017**	0.0074	2.3026	10	Lebron James.txt (9)
#19	champions	**0.0165**	0.0103	1.6094	12	Thomas Müller.txt (10)
#20	irish	**0.0153**	0.0067	2.3026	9	Lebron James.txt (9)
#21	istanbul	**0.0149**	0.0065	2.3026	6	Recep Tayyip Erdogan.txt (1)

Figure 4: Statistical information of the documents

stringdist The *stringdist*-package (van der Loo, 2014) implements distance calculation methods, like `Cosine`, `Jaccard`, `OSA` and other. We implemented functionality for calculating sentence similarities and provide an interactive visual representation. Each node represents an sentence of the selected document and the links between them represent the similarity of those sentences. The thicker the links, the more similar they are. By interacting with the visualization, corresponding sections of the document panel are getting highlighted, to see the similar sentences (see Figure 5). The bidirectional interaction functionality enables easy comparability.

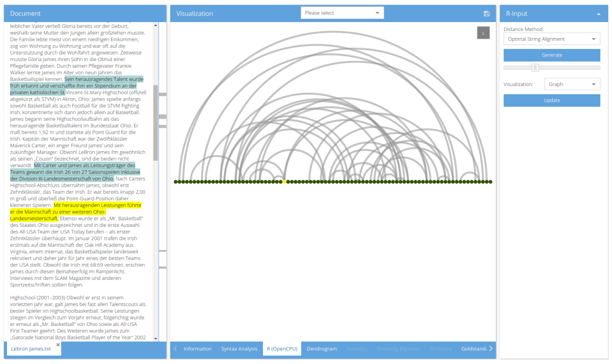


Figure 5: Depiction of sentence similarities.

stats We used functions from the *R*-package *stats* (R Development Core Team, 2008) to calculate a hierarchical cluster analysis based on the sentence similarities. This allows us to cluster similar sentences and visualize them with an interactive dendrogram. In Figure 6 we selected on one of these clusters and the document panel immediately adapts and highlights all the sentences in this cluster.

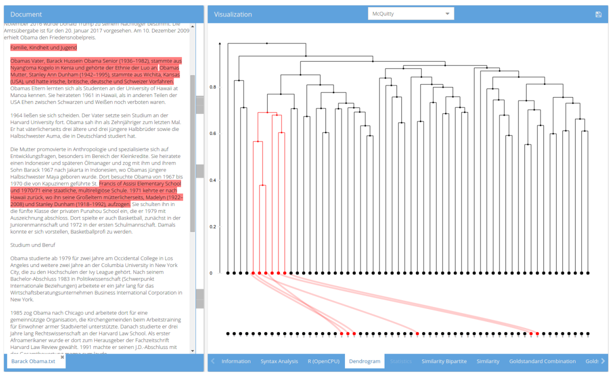


Figure 6: Similarity-clustered sentences.

An interesting side effect of integrating these tools into TextImager's pipeline is that their output can be concerted in a way to arrive at higher-level text annotations and analyses. In this way, we provide to an integration of two heavily expanding areas, that is, NLP and statistical modeling.

4 Future work

In already ongoing work, we focus on big data as exemplified by the Wikipedia. We also extend the number of built-in *R*-packages in *TextImager*.

5 Scope of the Software Demonstration

The beta version of *TextImager* is online. To test the functionalities of *R* as integrated into *TextImager* use the following demo: `http://textimager.hucompute.org/index.html?viewport=demo&file=R-Demo.xml`.

References

Jonathan Chang, 2015. *lda: Collapsed Gibbs Sampling Methods for Topic Models*. R package version 1.4.2.

Gabor Csardi and Tamas Nepusz. 2006. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695.

Maciej Eder, Jan Rybicki, and Mike Kestemont. 2016. Stylometry with R: a package for computational text analysis. *R Journal*, 8(1):107–121.

Ingo Feinerer and Kurt Hornik, 2015. *tm: Text Mining Package*. R package version 0.6-2.

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.

Fritz Günther, Carolin Dudschig, and Barbara Kaup. 2015. Lsafun: An R package for computations based on latent semantic analysis. *Behavior Research Methods*, 47(4):930–944.

Wahed Hemati, Tolga Uslu, and Alexander Mehler. 2016. Textimager: a distributed UIMA-based system for NLP. In *Proceedings of the COLING 2016 System Demonstrations*.

Jeroen Ooms. 2014. The OpenCPU system: Towards a universal interface for scientific computing through separation of concerns. *arXiv:1406.4806 [stat.CO]*.

R Development Core Team, 2008. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Julia Silge and David Robinson. 2016. tidytext: Text mining and analysis using tidy data principles in R. *JOSS*, 1(3).

M.P.J. van der Loo. 2014. The stringdist package for approximate string matching. *The R Journal*, 6:111–122.

Hadley Wickham. 2014. Tidy data. *Journal of Statistical Software*, 59(1):1–23.

Fridolin Wild, 2015. *lsa: Latent Semantic Analysis*. R package version 0.73.1.