UIMA-based JCoRe 2.0 Goes GitHub and Maven Central — State-of-the-Art Software Resource Engineering and Distribution of NLP Pipelines

Udo Hahn[1] **Franz Matthies**[1] **Erik Faessler**[1] **Johannes Hellrich**[1 2]
[1] Jena University Language & Information Engineering (JULIE) Lab
[2] Research Training Group "The Romantic Model. Variation - Scope - Relevance"
Friedrich-Schiller-Universität Jena
Jena, Germany
http://www.julielab.de

Abstract

We introduce JCoRe 2.0, the relaunch of a UIMA-based open software repository for full-scale natural language processing originating from the *Jena University Language & Information Engineering (*JULIE*) Lab*. In an attempt to put the new release of JCoRe on firm software engineering ground, we uploaded it to GitHub, a social coding platform, with an underlying source code versioning system and various means to support collaboration for software development and code modification management. In order to automate the builds of complex NLP pipelines and properly represent and track dependencies of the underlying Java code, we incorporated Maven as part of our software configuration management efforts. In the meantime, we have deployed our artifacts on Maven Central, as well. JCoRe 2.0 offers a broad range of text analytics functionality (mostly) for English-language scientific abstracts and full-text articles, especially from the life sciences domain.

Keywords: natural language processing pipelines, open software repository, software engineering, software configuration management, collaborative code management, JCoRe, UIMA, GitHub, Maven Central

1. Introduction

A true sign of the growing maturity of natural language processing is the continuous production of reusable code (e.g., for POS tagging, parsing, WSD, reference resolution, etc.) and language data resources (e.g., annotated corpora or computational lexicons). For a long time, a common way to share such artifacts has been to create a publically accessible space on one's own site, list available resources, and, after acknowledgment of licencing conditions, allow download of either source code or binaries.

We have been following this *passive directory model* for many years with our own NLP tool suite, the JULIE Lab UIMA Component Repository (JCoRe; Hahn et al. (2008)), as well. JCoRe – based on the UIMA middleware framework[1] – assembled (following UIMA speak) several collection readers (for document input), analysis engines (for NLP core tasks, e.g., document and sentence segmentation, named entity tagging or acronym resolution) and CAS consumers (e.g., for storing analysis results in an index structure). These resources were hosted and made publically accessible via http://www.julielab.de/Resources/JCoRe+NLP+Tools.html.

This service for the NLP community, despite the positive feedback we got from our users, increasingly also generated several problems which can be summarized as follows:

- Some modules maintained on an irregular basis became outdated (e.g., the MEDLINE[2] reader needs adaptation to the yearly updated MEDLINE XML schema),
- Bug fixes from external users or functional extensions they asked for were hard to integrate in the daily workflow. This is mainly due to the change patterns of the personnel working in academic software labs (e.g., the original developers of a piece of software leave after having completed their Master or Doctoral thesis, new staff entering usually focuses on novel goals rather than caring much about legacy software from former students). Hence, responding adequately to such feedback was not always easy to manage,
- Due to the lack of an automated build and release process, the tools on our web page had to be updated manually. This could lead to large discrepancies between the latest state of the source code in our private version control systems and the version published on JULIE Lab's web site. Hence, the integration of external feedback was even more impeded by the fact that suggestions from external users could refer to obsolete versions of our code,
- A communication bottleneck occurred on our lab's side since a growing number of users contacted us for support (e.g., concerning installation problems),
- The UIMA framework on which JCoRe is based often posed problems for users of our resources not so (well) acquainted with this environment. Hence, some UIMA tutoring and consulting was necessary for running JCoRe tools outside the JULIE Lab,
- Assembling a pipeline of JCoRe components required downloading the PEAR packages, installing

[1] https://uima.apache.org/

[2] MEDLINE is a bibliographical database for life sciences documents which currently comprises more than 25M references; for more details, cf. https://www.nlm.nih.gov/bsd/pmresources.html

them locally using the UIMA PEAR installer and then creating a UIMA pipeline descriptor. Since PEAR packages are installed in local directories, including library dependencies, such a pipeline could not easily be shared, e.g., by collaborators within the same group.

Other well-known representatives in the NLP domain who also adhere to the passive model for software distribution outlined above are, for instance

- LINGPIPE: http://alias-i.com/lingpipe
- GATE:[3] https://gate.ac.uk/

Despite the progress revealed by such repositories, from a software development perspective some fundamental shortcomings are evident. First, the interaction between producers and consumers of software housed in such repositories is fundamentally asymmetric and non-interactive. When the modules of choice have been downloaded from such sites they can be used on an 'as is' basis. But if bug fixes or reasonable extensions have been carried out at external sites, they have to be reported back informally, usually via email, to the original developers in a 'private' exchange mode. Also the channels for such software change communication are entirely decoupled from the platforms on which software development takes place. Furthermore, often no routine software engineering support (for version tracking, code merging, testing of modified code, etc.) is provided for professionally administering the proposed changes of the sources at the origin. So, embedded collaboration support is a clear desideratum.

Within the field of software engineering, team support for software developers has increasingly become a major concern (Mistrík et al., 2010). The growing relevance of social media for team building and collaborative, communication-intense group work has further fueled the emergence of so-called *social coding platforms* (for a survey of major players, cf. Begel et al. (2013)). One of the most prominent exemplars of this breed of software development frameworks is GITHUB[4] (Dabbish et al., 2012). With GITHUB, users are in command of a large-scale workspace where they place their software in repositories which are managed by a powerful version management system. Different change management policies and collaboration methods for the communication, notification (activity awareness and tracing) and visualization of source code changes on the basis of opted-in social roles (followers, watchers, etc.) are at the heart of GITHUB, thus adapting the social network metaphor to software development.

The enhanced opportunities and increased productivity gained by this and other comparable novel software development frameworks created entirely new communities of practice. This development had also a strong appeal for the NLP community. Many groups who once adhered to the passive directory model changed their rules of the game and subscribed to this collaborative *social coding model* in the NLP domain, for instance

- STANFORD TOOLS: https://github.com/stanfordnlp/CoreNLP, the primary site sits on GITHUB
- DKPRO CORE: https://github.com/dkpro/dkpro-core, the primary site sits on GITHUB
- NLTK: https://github.com/nltk/nltk, the primary site sits on GITHUB
- OPENNLP: https://github.com/apache/opennlp, GITHUB repository is a mirror of an SVN repository – their source code is located at https://svn.apache.org/repos/asf/opennlp/trunk/
- UIMA: https://github.com/apache/uima-uimaj, GITHUB repository is a mirror of an SVN repository – their source code is located at http://svn.apache.org/repos/asf/uima/addons/trunk/

As some application fields pose highly specific challenges for NLP pipelines (e.g., anonymization in the medical arena, gene or chemical name recognition in the fields of biology and chemistry, respectively), even for such specialized areas comprehensive NLP portals have been created—originally under the passive directory model, now increasingly moving to GITHUB as well, such as

- CTAKES (for medicine): https://github.com/apache/ctakes; (currently) seems incomplete on GITHUB, compared with the mirror at the project's web site under http://ctakes/apache.org/
- JCORE (for biology): http://julielab.github.io; the older version of JCORE maintained under http://www.julielab.de/Resources/JCoRe+NLP+Tools.html has become obsolete with the current publication of JCORE 2.0 on GITHUB.

Social interaction offers lots of unprecedented opportunities for efficient and effective coding. But keeping track of intrinsic inter-module dependencies and not losing control of versatile changes in complex software architectures introduces yet another orthogonal dimension of organizational complexities into distributed coding processes.

To properly document and lucidly track dependencies between pieces of code and even allow to automate the building of large-scale software projects, MAVEN[5] has turned out as a convenient build management and comprehension tool centered around the concept of a project object model (POM). To fully exploit the usage of MAVEN for JCORE, the deployment of our software to MAVEN CENTRAL has become the second software engineering cornerstone of the release of JCORE 2.0. Interestingly, with the exception of LINGPIPE and NLTK,[6] all other major

[3] The developers of GATE, in the meantime, provide their components via MAVEN CENTRAL.

[4] https://github.com/

[5] https://maven.apache.org/

[6] NLTK is a PYTHON project incompatible with MAVEN's dependence on JAVA. Yet NLTK has managed to integrate their framework in a software management tool similar to MAVEN.

NLP tool suites (i.e., STANFORD TOOLS, DKPRO CORE, OPENNLP, UIMA) have also been moving to deploy their software to MAVEN CENTRAL.

2. JCORE 2.0 — JULIE Component Repository

With JCORE 2.0, we relaunch the JULIE Component Repository in conformance with contemporary software development and engineering standards. This step is intended to not only simplify the accessibility and usability of our tools, but also implies that we commit ourselves to more rigid and frequent release cycles. JCORE 2.0 features, at the time of this writing (March 2016), 25 components (see Table 1), all written in JAVA. They are either self-developed JULIE Lab software or wrappers for third-party tools which we made compatible with UIMA and, more specific, with our type system (Buyko and Hahn, 2008). NLP pipelines within the UIMA framework we subscribed to can be configured from the following types of components:

- groups of several *collection readers* (CR) which enable users to access annotation information from other projects and corpora,
- several *analysis engines* (AE) which constitute the main part of an NLP pipeline ranging from low-level tasks such as tokenization to high-end functionality (e.g., dependency parsing, relation extraction, etc.),
- a couple of *CAS consumers* (CC) which export the annotations to different formats, and
- last but not least, the JULIE Lab *type system* (TS) which forms a comprehensive annotation type definition scheme.

Table 1 lists all components with a short description of their functionality and, if applicable, a reference to the description of the underlying module. Further information will be accessible from their respective GITHUB entry or their POM file. Compared with our initial JCORE repository (Hahn et al., 2008), some components are missing. These are either obsolete (e.g., the CAS2DB Consumer) or obtainable from other sources, as in the case of the LUCENE INDEXER which is now featured in the UIMA SANDBOX[7] (Faessler et al., 2009). For the majority of our AEs, we derived special packages consisting of the base component, a pre-trained model and a fitting UIMA component descriptor for 'out of the box' use.[8] These packages are bundled as JCORE *Projects* and also available on MAVEN CENTRAL and GITHUB. In Section 5., we will provide a short documentation on how to best use our newly packaged repository.

As already mentioned, we decided to tackle the problems of lacking visibility, collaboration, reuse and availability of JCORE by moving the tool suite to an open development framework. We committed ourselves to GITHUB in order to provide a platform for an unobstructed and interactive software exchange and MAVEN CENTRAL[9] for giving users of JCORE 2.0 the capability to access all of its components without much impediments. In this manner, we pave the way to easily plug together NLP pipelines without the need to worry much about UIMA technicalities.

Thus, the goal of making JCORE 2.0 tools public was, and still is, not only to provide a comfortable setting for our own work, but, first and foremost, to offer the scientific community easy to follow workflows to partake of results of cutting-edge research conducted in our lab. For instance, the newly added JULIE Lab part-of-speech tagger (JPOS) outperformed both the OPENNLP and the Stanford POS Tagger with regard to POS tagging in the German medical and newspaper domain (Hellrich et al., 2015).

As this paper describes the relaunch of an already acknowledged component repository with focus on its new paradigm, in the following, we will only give a concise recap of the different categories our components are divided into, with some general remarks about the workings of the latter. If needed, a more precise description can be found in Hahn et al. (2008) or the respective GITHUB page.

2.1. Type System

The data structure backbone of JCORE is still our comprehensive annotation type system (Buyko and Hahn, 2008). It offers a broad range of types to use in the context of various text analytics task—on a basic annotation level linguistic types like *sentence*, *token*, *abbreviation*, part-of-speech types, etc., or on a formal document structure level types such as *title*, *abstract*, *paragraph*, etc. Moreover the type system supports a substantial semantic layer for the biomedical domain including entities (such as *gene*, *organism*, *cell*), relations and events (e.g., various forms of *protein-protein interactions*). As this type system was already very elaborate and covered a large variety of morpho-syntactic and semantic features, there was not much to add except for some restructuring and minor extensions.

2.2. Collection Readers

JCORE 2.0 features six different collection readers which can be thought of as preprocessors for the actual text analysis tasks. Four of them (the ACE, BIONLP ST, IEXML/MANTRA and MUC7 Reader) comply with specific task-dependent file formats to feed the UIMA pipeline with semantically annotated text from the respective shared task—from the biomedical domain (BIONLP ST and MANTRA) and the newswire domain (ACE and MUC7).[10] The other two (XML and FILE Reader) are more general in nature and read XML files or plain text files, respectively. The former reader is extensible by a mapping file to conform to specific formats (e.g. MEDLINE or PUBMED; cf. Footnote 2).

2.3. Analysis Engines

The analysis engines are at the heart of every UIMA pipeline and are (in the case of JCORE) responsible for

[7] https://uima.apache.org/sandbox.html#lucas.consumer

[8] AEs for which such a package exists are marked accordingly with '*' in Table 1.

[9] http://search.maven.org/

[10] However, the BIONLP ST corpora are the only ones that are freely available.

Component	Type	Functional Description, Including Specification of Sources and References
JULIE Type System	TS	Annotation type system with an extensive semantic layer (Buyko and Hahn, 2008)
ACE Reader	CR	Reader which converts the ACE (Automatic Content Extraction) corpus (https://www.ldc.upenn.edu/collaborations/past-projects/ace) (Doddington et al., 2004) to CAS objects
MUC7 Reader	CR	Reader which converts MUC-7 (Message Understanding Conference) files (https://catalog.ldc.upenn.edu/LDC2001T02) (Chinchor, 1998) to CAS objects
BIONLP ST Reader	CR	Reader which converts BIONLP Shared Task formatted files (http://www.nactem.ac.uk/tsujii/GENIA/SharedTask/index.shtml\#data) (Kim et al., 2009) to CAS objects
IEXML / MANTRA Reader	CR	Reader for IEXML files as used in the MANTRA Challenge (https://sites.google.com/site/mantraeu/clef-er-challenge) (Hellrich et al., 2014)
*XML Reader	CR	Reader which employs a mapping file for reading, e.g., PUBMED & MEDLINE files (http://www.ncbi.nlm.nih.gov/pubmed)
FILE Reader	CR	Reader which takes plain text files as input
*JULIE Lab Sentence Splitter	AE	A CRF-based sentence splitter (Tomanek et al., 2007)
*OPENNLP Sentence Splitter	AE	Wrapper for OPENNLP's sentence splitter (https://github.com/apache/opennlp)
*JULIE Lab Tokenizer	AE	A CRF-based tokenizer (Tomanek et al., 2007)
*OPENNLP Tokenizer	AE	Wrapper for OPENNLP's tokenizer (https://github.com/apache/opennlp)
STANFORD Lemmatizer	AE	Wrapper for the Stanford Lemmatizer which yields morphological normalization, i.e. a mapping from inflected word forms to the associated lemma (https://github.com/stanfordnlp/CoreNLP)
*JULIE Lab POS Tagger	AE	Tagger for the annotation of part-of-speech tags from an arbitrarily chosen tag set (Hellrich et al., 2015)
*OPENNLP POS Tagger	AE	Wrapper for OPENNLP's POS tagger (https://github.com/apache/opennlp)
*OPENNLP Chunker	AE	Wrapper for OPENNLP's text chunker (https://github.com/apache/opennlp)
*MST Dependency Parser	AE	Wrapper for the MST dependency parser (McDonald et al., 2005)
*OPENNLP Constituency Parser	AE	Wrapper for OPENNLP's shift-reduce parser (https://github.com/apache/opennlp)
Acronym Resolver	AE	System for the resolution of acronyms (short form → long form) (Schwartz and Hearst, 2003)
LINGPIPE Gazetteer	AE	Wrapper for LINGPIPE's gazetteer (http://alias-i.com/lingpipe/)
*JULIE Lab Named Entity Tagger	AE	JNET, a tagger for the automatic detection and classification of named entity mentions in running text (Hahn et al., 2008)
JULIE Lab Coordination Resolver	AE	Tagger for the recognition and resolution of coordinated elliptical entity expressions (Buyko et al., 2007)
*BIOSEM Relation Extractor	AE	Wrapper for the BIOSEM Event Extraction System (Bui and Sloot, 2012)
BIONLP ST Consumer	CC	Consumer which writes CAS annotations into the BIONLP Shared Task format
CAS2IOB Consumer	CC	Consumer which generates IOB-formatted files for specified annotations
XMI Writer	CC	Wrapper for UIMA's XMI writer, with some additional options
IEXML / MANTRA Consumer	CC	Consumer which generates stand-off IEXML files as used in the MANTRA Challenge (https://sites.google.com/site/mantraeu/clef-er-challenge) (Hellrich et al., 2014)

Table 1: Overview of the JCORE 2.0 Component Repository

the actual text processing. JCORE 2.0 contains at the time of this writing 15 AEs which deal with either morpho-syntactic or semantic processing.

Morpho-Syntactic Processing. Token and sentence segmentation is taken care of by a wrapper for the OPENNLP tool suite based on Maximum Entropy (ME) models (Berger et al., 1996) and a self developed tool based on Conditional Random Fields (CRF) (Lafferty et al., 2001). In order to deal with morphological variation of words, we provide the Stanford Lemmatizer. This component takes at least tokenized and POS-tagged text and returns a dictionary form of each word. To provide the aforementioned part-of-speech (POS) tags, we supply a wrapper for OPENNLP's POS tagger and a self developed component (Hellrich et al., 2015). The tool set for syntactic analysis features a phrase chunker, a constituency parser (both wrappers for OPENNLP) and a slightly modified version of the MSTPARSER (McDonald et al., 2005), a parser for non-projective dependency structures, where we adapted the source code to UIMA's workflow so that models are only loaded once in the course of the initialization phase.

Semantic Processing. Acronym resolution (the task of

finding a full form for a corresponding short form or abbreviation, such as for *UN* → *United Nations*) is supported by our reimplementation of the Schwartz-Hearst algorithm (Schwartz and Hearst, 2003). Named entities are handled by two different tools: one is based on LINGPIPE's gazetteer and the other is JNET, a CRF-based entity tagger developed at JULIE Lab that can be used for arbitrary domains and entity classes given appropriate training material (Hahn et al., 2008). Thirdly, for the recognition and resolution of coordinated elliptical entity expressions we currently only provide a Lab-made rule-based tagger. A machine learning-based tagger became nonfunctional as a result of switching to new versions because of global dependencies in JCORE 2.0 and, thus, needs a major overhaul.

Finally, as JCORE's top level analysis component, we integrated the BIOSEM relation extractor (Bui and Sloot, 2012) into our repository. It not only achieves state-of-the-art performance in tasks dealing with the extraction of events and relations in the biomedical domain,[11] but also excels with truly competitive compute-time performance data when compared to relation extractors which use dependency parsing, as well as with small-sized models.

2.4. CAS Consumers

The CAS consumers are the post-processors of a UIMA pipeline and deploy the results of the AEs in different formats depending on the actual consumer. These formats could either be text/XML files, databases or, like in the case of the LUCENE INDEXER (Faessler et al., 2009), even more sophisticated outputs. JCORE consists of four different consumers, two of which are more specific and accompany the corresponding CRs, namely the BIONLP ST Consumer and the IEXML/MANTRA Consumer which generate files expected by the appropriate task. The XMI Writer is a more complex wrapper around the UIMA inherent XMICASSERIALIZER as it also allows single or multiple files to be compressed into zip files. The CAS2IOB Writer produces IOB-formatted text files and can be adjusted to limit the output to specific annotation types only.

As already mentioned the CAS2DBCONSUMER fell victim to our updating process, as many dependencies need to be restructured to fit into the new JCORE scheme. Its revitalization is on the agenda for future versions of JCORE.

3. GITHUB

In the past years, GITHUB has become the most popular Web-based social code sharing service world-wide.[12] Based on the GIT distributed version control system,[13] it has emerged as an essential tool in technology areas that require intense human group collaboration for effective task completion, such as software development, technical and business co-authoring, education, etc. (Begel et al., 2013).

GITHUB's positive impact on such collaboration processes (as reported, e.g., by Tsay et al. (2012)) is mainly due to the awareness and transparency features it provides to team, project and community members (Dabbish et al., 2012).

The following description, quoted from Zagalsky et al. (2015), nicely summarizes the main features of GITHUB (for another brief summary, cf. Dabbish et al. (2012)[p.1280]). It "offers several unique features to facilitate user collaboration. Its most important feature is the *Pull Request* (PR) mechanism which is a way to initiate discussion with other users and share or comment on the various artifacts in a project (typically changes to the project's content). The discussion may include code that is visible to everyone and it shows the exact changes that would be merged if the PR were accepted. A PR may involve other content (e.g., screenshots) to provide a background for the discussion, or include changes to other resources in the project.

When a user wishes to contribute to someone else's project, they can *Clone*[14] the project to create a full copy of the project in their local environment, but where committed changes will still affect the original project. This is called a *Shared Repository Model*:[15] contributors can either commit changes directly into the shared repository or use PRs to start code reviews and conversations about proposed changes before the changes are merged into the master branch.

Alternatively, a user can *Fork*[16] the entire project to create a parallel project where committed changes do not directly affect the original project. This is called a *Fork & Pull Model*: PRs provide a way to notify the original project maintainers about the changes you would like them to consider.

Users can not only follow other users or projects of interest, but they can also broadcast their activities to their followers. Furthermore, users can discover new projects by using the *Explore* feature, or share snippets using the *Gist* feature. GITHUB also supports awareness by broadcasting updates to the user's news feed. The combination of these features facilitates *"a culture of spontaneous-but-structured collaboration"*."[17] (Zagalsky et al., 2015)[p.1907-08].

4. Maven

MAVEN, the widely used JAVA dependency and build management tool, communicates and interacts by default with *The Central Repository* which offers the largest collection of JAVA components.[18] This forms the already mentioned MAVEN CENTRAL, where we will deploy all JCORE 2.0 components and their dependencies; nearly all of the dependencies that were not developed by us are already

[11] In the *BioNLP 2013 Shared Task*, the system ranked 3rd and 1st regarding approximate and strict matching, respectively. In the *2011 Shared Task* it ranked 1st for full texts and 3rd for abstracts.

[12] As of March 2016, there are 12M people collaborating across 31M repositories on GITHUB; see `https://github.com/about/press`

[13] `https://git-scm.com/`

[14] `https://help.github.com/articles/duplicating-arepository`

[15] `https://guides.github.com/introduction/flow/`

[16] `https://help.github.com/articles/fork-a-repo`

[17] `tp://software-carpentry.org/blog/2012/04/github-for-education.html`

[18] `http://central.sonatype.org/pages/about.html`

present there.
To facilitate a uniform build system, each MAVEN project features as its core a POM file which is the pivotal point in the format of an XML file that not only specifies project metadata but also handles configuration, e.g., resolving dependencies. This might arguably be one of the main strengths of MAVEN, in general. It empowers the user to simply declare what other components are needed for a project and the appropriate libraries are automatically and dynamically loaded into the project's building path, either from the *Local* or the *Remote Repository*,[19] if they are non-existent in the former or newer in the latter. This results in a plethora of well-maintained and up-to-date libraries all of them accessible in an comprehensible way. For well-known and frequently used JAVA Integrated Development Environments (IDEs, such as ECLIPSE, NETBEANS or INTELLI IDEA), there also exist MAVEN plug-ins that make the preparation and modification of POMs even more handy.

5. JCoRe 2.0—How to

Naturally, with the transition to JCoRe 2.0 and the farewell from a passive model of software development and distribution, we refrained from providing PEAR packages. There are now basically two ways of using the components of our repository to plug together NLP pipelines which we will describe, in brevity, in the following.[20] The JCoRe *Pipelines Repository* contains projects that exemplify both strategies. Regardless of which approach is chosen, one needs to operate with JAVA, MAVEN and UIMA.

DECLARATIVE. On the one hand, there is the declarative approach where one utilizes a so-called *Collection Processing Engine* (CPE), an XML file which declares which components and respective settings to use. At the time of this writing, we are working on a reference sheet where the potential user will find the coordinates for each component to fill out these CPEs and scripts that will provide an easy way to prepare the pipeline (e.g., checking if all is set up correctly and downloading the appropriate MAVEN artifacts). Using, understanding and/or modifying code is not necessary for this method.

PROCEDURAL. On the other hand, a more programmatic approach is to employ, for instance, UIMA FIT[21] in order to simplify the description and instantiation of components and write the actual pipeline in JAVA code. This variant offers its users a conspicuous level of control over the pipeline. As with the former procedure, MAVEN takes care of the dependency resolution and loads the necessary libraries dynamically into the workspace.
This technique leads to an even more proactive mode, if one is willing to collaborate in further development and/or bug-fixing of our components by downloading the up-to-date GITHUB repository and use its content to directly assess any change to a component one has made.

To complete this section, we will give a concise overview of the different relevant repositories into which we have split up JCoRe 2.0 on GITHUB:

JCoRe Base. The *Base* package contains all basic components—the Type System, all CRs, AEs and CCs. This repository is highly relevant if one is interested in active development of JCoRe. All projects included come only with a basic descriptor file and are rarely usable on their own since they lack model files.

JCoRe Projects. The *Project* package covers pre-built projects that should be used if one hasn't the need to train models on one's own. The components of this repository don't feature actual code but rather consist of a POM file with coordinates to their respective BASE package, a fully functional descriptor XML and a model from the biomedical domain. Every BASE component which needs training is featured at least once here.

JCoRe Pipelines. In the *Pipeline* package we provide pre-built pipelines for specific NLP tasks. Each individual component comes at least with the scripts mentioned beforehand, a CPE and a POM file. At the time of this writing (March 2016), it contains a named entity detection pipeline that takes simple text files as input and returns IOB-formatted files with annotations for entities as output; this pipeline uses the CPE approach. The other component utilizes the BIOSEM relation extractor to read text and protein files from the *BioNLP Shared Task* in order to produce appropriate event annotation files with a model trained on the 2011 Shared Task data.

6. Conclusion

In this paper, we described the transition from the initial JCoRe repository release which relied on a passive model of software distribution to JCoRe 2.0 which features a social coding model. This move expresses a clear commitment to contemporary standards of an open and collaborative software management model as promoted by an increasing number of leader groups in the NLP community. As JCoRe 2.0 mainly aims to provide tools for the field of biomedical NLP, we believe to have made a major step towards reusability, accessibility and collaboration in this area.
The workflow of initializing adjustments to and modifications of our components is made even easier and can now proceed in an interactive manner. Meaningful and beneficial changes by the community can make it more swiftly into our tool set, as we envisage more frequent release cycles. Even if potential users of JCoRe 2.0 are only interested in using our components 'as is', the migration to MAVEN CENTRAL supplies better and easier access paths.

Availability: The most convenient way to get an overview of and access to the source code of JULIE Lab's

[19] Simply speaking, a *Remote Repository* is, for instance, the aforementioned *Central Repository* but can also be set up and specified to be on a private server. The *Local Repository* is a cache of the downloads from the former.

[20] Some components have a standalone mode, as well. How to use them in this manner will be explained on their respective GITHUB page.

[21] `https://uima.apache.org/uimafit.html`

JCoRe components is by following the GitHub Page `http://julielab.github.io`

7. Acknowledgments

This work is partially supported by grants from the *Deutsche Forschungsgemeinschaft (DFG)*—Erik Faessler is funded by a grant within the CRC AquaDiva (SFB 1076), while Johannes Hellrich is funded by a grant from the *Research Training Group "The Romantic Model. Variation - Scope - Relevance"* (GRK 2041/1).

8. Bibliographical References

Begel, A., Bosch, J., and Storey, M.-A. (2013). Social networking meets software development: Perspectives from GitHub, MSDN, Stack Exchange, and TopCoder. *IEEE Software*, 30(1):52–66.

Berger, A. L., Della Pietra, S. A., and Della Pietra, V. J. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Bui, Q.-C. and Sloot, P. M. A. (2012). A robust approach to extract biomedical events from literature. *Bioinformatics*, 28(20):2654–2661.

Buyko, E. and Hahn, U. (2008). Fully embedded type systems for the semantic annotation layer. In *ICGL 2008 — Proceedings of the 1st International Conference on Global Interoperability for Language Resources. Hong Kong, SAR, January 9-11, 2008*, pages 26–33.

Buyko, E., Tomanek, K., and Hahn, U. (2007). Resolution of coordination ellipses in biological named entities using Conditional Random Fields. In *PACLING '07 — Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics. Melbourne, Australia, September 19-21, 2007*, pages 163–171.

Chinchor, N. A. (1998). Overview of Muc-7/Met-2. In *MUC-7 — Proceedings of the 7th Message Understanding Conference. Fairfax, Virginia, USA, April 29 - May 1, 1998*.

Dabbish, L., Stuart, C., Tsay, J. T., and Herbsleb, J. (2012). Social coding in GitHub: Transparency and collaboration in an open software repository. In *CSCW '12 — Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work. Seattle, WA, USA, February 11-15, 2012*, pages 1277–1286.

Doddington, G., Mitchell, A., Przybocki, M., Ramshaw, L. A., Strassel, S., and Weischedel, R. M. (2004). The Automatic Content Extraction Ace Program: Tasks, data & evaluation. In *LREC 2004 — Proceedings of the 4th International Conference on Language Resources and Evaluation. In Memory of Antonio Zampolli. Lisbon, Portugal, 24-30 May, 2004*, volume 3, pages 837–840.

Faessler, E., Landefeld, R., Tomanek, K., and Hahn, U. (2009). LuCas: - a Lucene CAS Indexer. In Christian Chiarcos, et al., editors, *Von der Form zur Bedeutung: Texte automatisch verarbeiten. From Form to Meaning: Processing Texts Automatically. Proceedings of the Biennial GSCL Conference 2009*, pages 217–224, Tübingen. Gunter Narr Verlag.

Hahn, U., Buyko, E., Landefeld, R., Mühlhausen, M., Poprat, M., Tomanek, K., and Wermter, J. (2008). An overview of JCoRe, the Julie Lab UIMA Component Repository. In *Proceedings of the LREC '08 Workshop "Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP". Marrakech, Morocco, 31 May 2008*, pages 1–7.

Hellrich, J., Clematide, S., Hahn, U., and Rebholz-Schuhmann, D. (2014). Collaboratively annotating multilingual parallel corpora in the biomedical domain: Some Mantras. In *LREC 2014 — Proceedings of the 9th International Conference on Language Resources and Evaluation. Reykjavik, Iceland, May 26-31, 2014*, pages 4033–4040.

Hellrich, J., Matthies, F., Faessler, E., and Hahn, U. (2015). Sharing models and tools for processing German clinical texts. In Ronald Cornet, et al., editors, *Digital Healthcare Empowering Europeans. Proceedings of the 26th Medical Informatics in Europe Conference — MIE 2015. Madrid, Spain, May 27-29, 2015*, number 210 in Studies in Health Technology and Informatics, pages 734 – 738, Amsterdam etc. IOS Press.

Kim, J.-D., Ohta, T., Pyysalo, S., Kano, Y., and Tsujii, J. (2009). Overview of BioNLP '09 Shared Task on Event Extraction. In *BioNLP 2009 — Proceedings of the Companion Volume: Shared Task on Event Extraction. Boulder, CO, USA, June 5, 2009*, pages 1–9.

Lafferty, J. D., McCallum, A. K., and Pereira, F. C. N. (2001). Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *ICML '01 — Proceedings of the 18th International Conference on Machine Learning. Williams College, Williamstown, MA, USA , June 28 - July 1, 2001*, pages 282–289.

McDonald, R. T., Pereira, F. C. N., Ribarov, K., and Hajič, J. (2005). Non-projective dependency parsing using spanning tree algorithms. In *HLT-EMNLP 2005 — Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing. Vancouver, B.C., Canada, October 6-8, 2005*, pages 523–530.

Ivan Mistrík, et al., editors. (2010). *Collaborative Software Engineering*. Springer-Verlag, Berlin, Heidelberg.

Schwartz, A. S. and Hearst, M. H. (2003). A simple algorithm for identifying abbreviation definitions in biomedical text. In Russ B. Altman, et al., editors, *PSB 2003 — Proceedings of the Pacific Symposium on Biocomputing 2003. Kauai, Hawaii, USA, January 3-7, 2003*, pages 451–462.

Tomanek, K., Wermter, J., and Hahn, U. (2007). Sentence and token splitting based on Conditional Random Fields. In *PACLING '07 — Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics. Melbourne, Australia, September 19-21, 2007*, pages 49–57.

Tsay, J. T., Dabbish, L., and Herbsleb, J. (2012). Social media and success in Open Source projects. In *CSCW '12 — Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work. Companion Volume. Seattle, WA, USA, February 11-15, 2012*, pages 223–226.

Zagalsky, A., Feliciano, J., Storey, M.-A., Zhao, Y., and Wang, W. (2015). The emergence of GITHUB as a collaborative platform for education. In *CSCW '15 — Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing. Vancouver, B.C,. Canada, March 14-18, 2015*, pages 1906–1917.