

Documentation for  
**JULIE Lab Token Boundary Detector**  
Version 2.1

Karin Tomanek  
Jena University Language & Information Engineering (JULIE) Lab  
Fürstengraben 30  
D-07743 Jena, Germany  
`katrin.tomanek@uni-jena.de`

## 1 Objective

The JULIE Lab Token Boundary Detector (UIMA-JTBD) is a token boundary detector for UIMA. It is part of the JULIE Lab NLP tool suite<sup>1</sup> which contains several NLP components (all UIMA compliant) from sentence splitting to named entity recognition and normalization as well as a comprehensive UIMA type system.

UIMA-JTBD is an UIMA wrapper for JTBD, the respective command-line version. For more detailed information on the functioning of JTBD check the JTBD documentation or refer to [TWH07].

## 2 Installation

UIMA-JTBD comes as a UIMA pear file. Run the Pear-Installer (e.g., `./runPearInstaller.sh` for Linux) from your UIMA-bin directory. After installation, you will find a subfolder `desc` in your installation folder. This directory contains a descriptor `TokenAnnotator.xml` for UIMA-JTBD. You may now e.g. run UIMA's Collection Proprocessing Engine Configurator (`cpeGUI.sh`) and add UIMA-JTBD as a component into your NLP pipeline.

This pear package also contains a model for tokenization splitting. The model was trained on a special bio-medical corpus which consists of data from (manually annotated) material which we took from MedLine abstracts and a modified version of PENNBIOIE's<sup>2</sup>

---

<sup>1</sup><http://www.julielab.de/>

<sup>2</sup><http://bioie.ldc.upenn.edu/>

underlying tokenization. In the PENNBIOIE corpus, some purely alphanumeric strings are divided into smaller tokens to support PENNBIOIE's entity annotation, especially in a common annotation for variation events (e.g. "S45F" with "S"=state\_original, "45"=location, "F"=state\_altered). Those splits were (manually) undone to fit our tokenization guidelines. Currently, our tokenization corpus comprises about 36000 sentences. An accuracy of ACC=96.7% is reached on this data using 10-fold cross-validation. You will find the model trained on this data in the directory **resources**.

### 3 Requirements and Dependencies

UIMA-JTBD is written in Java (version 1.5 or above required) using Apache UIMA version 2.2.1-incubation<sup>3</sup>.

The input and output of an AE takes place by annotation objects. The classes corresponding to these objects are part of the *JULIE Lab UIMA Type System* in its current version (2.1).<sup>4</sup>

UIMA-JTBD in its current version is based on JTBD-1.6.1 which employs the machine learning toolkit MALLET [McC02].

### 4 Using the AE – Descriptor Configuration

In UIMA, each component is configured by a descriptor in XML. In the following we describe how the descriptor required by this AE can be created with the *Component Descriptor Editor*, an Eclipse plugin which is part of the UIMA SDK.

A descriptor contains information on different aspects. The following subsection refers to each sub aspect of the descriptor which is, in the Component Descriptor Editor, a separate *tabbed page*. For an indepth description of the respective configuration aspects or tabs, please refer to the *UIMA SKD User's Guide*<sup>5</sup>, especially the chapter on "Component Descriptor Editor User's Guide".

To define your own descriptor go through each tabbed pages mentioned here, make your respective entries (especially in page *Parameter Settings* you will be able to configure JNET to your needs) and save the descriptor as **SomeName.xml**.

Otherwise, you can of course employ the descriptor that is contained in the pear package you downloaded (in your installation directory, see **desc/TokenAnnotator.xml**).

---

<sup>3</sup><http://incubator.apache.org/uima/>

<sup>4</sup>The *JULIE Lab UIMA type system* can be separately obtained from <http://www.julielab.de/>, however, this package already includes the necessary parts of the type system.

<sup>5</sup><http://incubator.apache.org/uima/>

**Overview** This tab provides general information about the component. For the UIMA-JTBD you need to provide the information as specified in Table 1.

Subsection	Key	Value
Implementation Details	Implementation Language	Java
	Engine Type	primitive
Runtime Information	updates the CAS	yes
	multiple deployment allowed	yes
	outputs new CASes	no
	Name of the Java class file	de.julielab.jules.ae.TokenAnnotator
Overall Identification Information	Name	Token Annotator
	Version	2.1
	Vendor	JULIE Lab
	Description	not needed

Table 1: Overview/General Settings for AE.

**Aggregate** Not needed here, as this AE is a primitive.

**Parameters** See Table 2 for a specification of the configuration parameters of this AE. Do not check “Use Parameter Groups” in this tab.

Parameter Name	Parameter Type	Mandatory	Multivalued	Description
ModelFilename	String	yes	no	filename to model trained for JTBD

Table 2: Parameters of this AE.

**Parameter Settings** The specific parameter settings are filled in here. For each of the parameters defined in 4, add the respective values here (has to be done at least for each parameter that is defined as mandatory). See Table 3 for the respective parameter settings of this AE.

**Type System** On this page, go to *Imported Type* and add the *JULIE UIMA Type System*. (Use “Import by Location”).

Parameter Name	Parameter Syntax	Example
ModelFilename	full path	resources/JULIE_life-science-1.6.mod.gz

Table 3: Parameter settings of this AE.

**Capabilities** The tokenizer takes as input annotations from type `de.julielab.jules.types.Sentence` and returns annotations from type `de.julielab.jules.types.Token`. See Table 4.

Type	Input	Output
<code>de.julielab.jules.types.Sentence</code>	✓	
<code>de.julielab.jules.types.Token</code>		✓

Table 4: Capabilities of this AE.

**Index** Nothing needs to be done here.

**Resources** Nothing needs to be done here.

## 5 Copyright and License

This software is Copyright (C) 2008 Jena University Language & Information Engineering Lab (Friedrich-Schiller University Jena, Germany), and is licensed under the terms of the Common Public License, Version 1.0 or (at your option) any subsequent version.

The license is approved by the Open Source Initiative, and is available from their website at <http://www.opensource.org>.

## References

- [McC02] Andrew McCallum. Mallet: A machine learning for language toolkit. <http://mallet.cs.umass.edu>, 2002.
- [TWH07] Katrin Tomanek, Joachim Wermter, and Udo Hahn. A reappraisal of sentence and token splitting for life science documents. In *MEDINFO 2007 - Proceedings of the 12th World Congress on Medical Informatics.*, 2007.