Rhetector: Rhetorical Entity Detector Component

Guide for Users and Developers

Bahar Sateli
René Witte

Release 1.0
August 10, 2015

Semantic Software Lab
Concordia University
Montréal, Canada
http://www.semanticsoftware.info

Contents

About this document

This document contains documentation for the *Rhetorical Entity Detection Component (Rhetector)*. You can obtain the latest version from http://www.semanticsoftware.info/rhetector.

License

Support

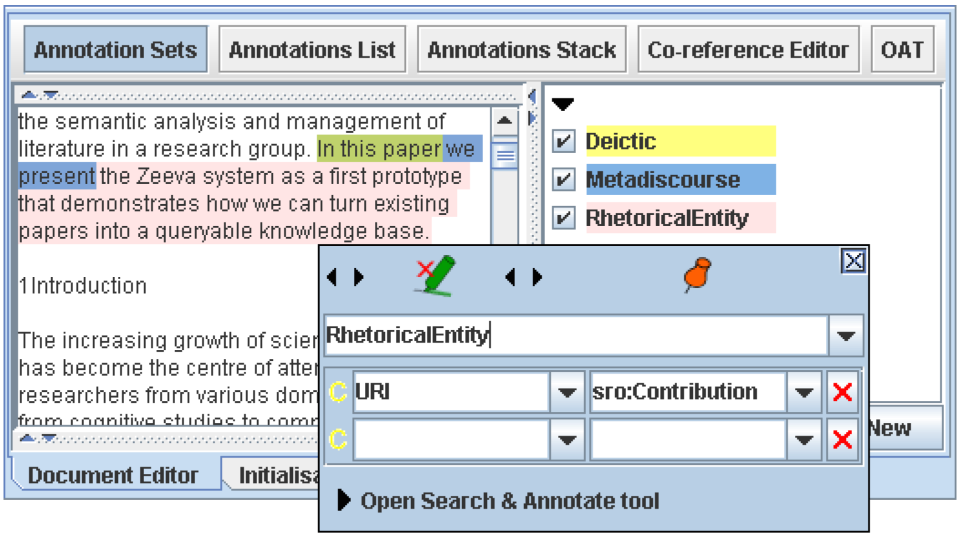
For question or general comments, please use the online forum at http://www.semanticsoftware.info/forums/tools-resources-forum/durm-corpus-wiki-tools

[1]LGPL3, https://www.gnu.org/licenses/lgpl-3.0.en.html

Chapter 1

Rhetorical Entity Detector (Rhetector)

Rhetector is a text mining component for the automatic extraction of Rhetorical Entities (REs) from scholarly literature, such as journal articles. Rhetector is implemented based on the GATE framework with various rules in the JAPE language.


the semantic analysis and management of literature in a research group. In this paper we present the Zeeva system as a first prototype that demonstrates how we can turn existing papers into a queryable knowledge base.

1 Introduction

The increasing growth of scie
has become the centre of atte
researchers from various dom
from cognitive studies to com

RhetoricalEntity

URI | sro:Contribution

Figure 1.1: Example annotations generated by Rhetector on a document

1.1 Overview

In the context of scientific literature, Rhetorical Entities (REs) are spans of text (sentences, passages, sections, etc.) in a document, where authors convey their findings, like Claims or Arguments, to the readers. REs are usually situated in certain parts of a document, depending on their role. For example, the authors' Claims are mentioned in the *Abstract*, *Introduction* or *Conclusion* section of a paper, and seldom in the *Background*. This conforms with the researchers' habit in both reading and writing scientific articles. Verbatim extraction of REs from text helps to efficiently allocate the attention of humans when reading a paper, as well as improving retrieval mechanisms by finding documents based on their REs (e.g., *"Give me all papers with implementation details"*).

We designed a lightweight pipeline to automatically detect rhetorical entities in scientific literature, currently limited to Claims and Contributions. It can classify sentences of a document into one of three categories (Claim, Contribution, or neither) using a rule-based approach.

For each detected RE, an annotation “RhetoricalEntity” is added to the document. Based on the grammatical structure of the RE, it is classified and mapped onto existing concepts on the Linked Open Data (LOD) cloud. The fully-qualified URI of the RE type is stored as the value of the “URI” feature of each annotation (see Figure 1.1).

1.2 Installation

Prerequisites. To run Rhetector, you must have GATE installed. Please refer to http://gate.ac.uk for instructions on how to download and install GATE. The automated install method described below requires GATE version 8 or newer.

Demo Pipeline. The Rhetector distribution comes with a demo pipeline that you load directly from within GATE:

1. Start GATE, open the Plugin Manager *(File → Manage CREOLE Plugins. . .)*
2. On the fourth tab *(Configuration)*, select a writable directory for the local plugin installations and enable the “Semantic Software Lab” repository
3. On the third tab *(Available to install)*, you will now find Rhetector. Select it and click on “Apply All”.
4. On the first tab *(Installed Plugins)*, you will now find the Rhetector plugin. Select it (either “Load Now” or “Load Always”) and click on “Apply All”.

To load the example pipeline, go to *File → Ready Made Applications → Semantic Software Lab → Rhetorical Entity Extraction Demo Application.*[1]

1.3 Usage

The Rhetector component provides two Processing Resources (PRs) that are pre-configured to load their corresponding Gazetteer and JAPE rules. Note that Rhetector needs the text to be pre-processed to function properly, for which you can load several PRs from the *ANNIE* and *Tools* plug-ins in the following order:

Document Reset PR resets any existing annotation in the document

ANNIE English Tokeniser breaks down the document content text into *Tokens*

RegEx Sentence Splitter aggregates word tokens into sentences

ANNIE POS Tagger adds a Part-of-Speech tag to each word token

GATE Morphological Analyser adds the root of each token as a feature

Finally, add the **Rhetorical Entity Gazetteer** and **Rhetorical Entity Transducer** PRs to end the of sequence and run the pipeline on your corpus.

[1] Please refer to the GATE user’s guide, http://gate.ac.uk/sale/tao for further details on working with pipelines in GATE.

1.4 Implementation notes

Rhetector is implemented as a set of grammars running a multi-phase JAPE transducer. The rules operate on the document's text tokens, their root features and *Lookup* tokens, generated by the **Rhetorical Entity Gazetteer** PR. The `main.jape` contains the transducer definition, which generally contains several phases:

deictics.jape creates *Deictic* annotations for expressions within an utterance that refer to parts of the discourse. For example, the word "here" in "here, we describe a new methodology. . . " refers to the article that the user is reading.

metadiscourse.jape uses a combination of deictic phrases and work tokens to create *Metadiscourse* annotations, i.e., phrases in sentences that provide a high-level overview of what is presented in the paper.

re_sentence.jape creates *RhetoricalEntity* annotations from sentences containing metadiscourse elements and classifies them based on their main verb.

1.5 Changelog

Version 1.0 (10.08.2015)

First public release.