

# The GATE Multi-Parser Predicate-Argument EXtractor Component (MultiPaX)

- [GATE Components](#)
- [NLP](#)
- [Text Mining](#)

toc\_collapse=0; Table of Contents















- [1. Overview](#)
- [2. Prerequisites](#)
- [3. Documentation](#)
- [4. Pipeline Configuration](#)
- [5. MultiPaX Configuration](#)
- [6. Result Annotations](#)
- [7. Downloads](#)
- [8. Feedback](#)
- [9. Version history](#)

## 1. Overview

This page describes a [GATE](#) component for extracting predicate-argument structures (PAS) from the output of different parsers. PASs are used in various contexts to represent relations within a sentence structure. Different "semantic" parsers extract relational information from sentences but there exists no common format to store these information. Our *Multi-Parser Predicate-Argument Extractor* component MultiPaX (formerly just named PAX) takes the annotations generated by selected parsers and extracts/transforms the parsers' results to predicate-argument structures represented as triples (subject-verb-object).

A variety of different parsers offer support for syntactic analysis of sentences. With this resource we try to extract predicate-argument structures using the output of different such parsers. Currently we support MiniPar, RASP, SUPPLE, and the Stanford Parser. In addition, we can extract PAS out of noun phrases, by making use of the output of a noun phrase chunker like [MuNPEX](#).

Our resource is implemented as a component for the [General Architecture for Text Engineering \(GATE\)](#). For a more detailed background and motivation, please read our research paper [1]: [Krestel, R., R. Witte, and S. Bergler](#), "[Predicate-Argument EXtractor \(PAX\)](#)", *New Challenges for NLP Frameworks*, Valletta, Malta : ELRA, pp. 51--54, May 22, 2010.

Selected Processing resources		
I	Name	Type
	Document Reset PR_0001A	Document Reset PR
	ANNIE English Tokeniser_00018	ANNIE English Tokeniser
	ANNIE Sentence Splitter_0001D	ANNIE Sentence Splitter
	ANNIE POS Tagger_00025	ANNIE POS Tagger
	GATE Morphological analyser_0003F	GATE Morphological analyser
	NPE	Jape Transducer
	Minipar Wrapper_00025	Minipar Wrapper
	SUPPLE Parser_00026	SUPPLE Parser
	StanfordParser_00027	StanfordParser
	MultiPaX_0001D	MultiPaX
	MultiPaX_0001D	MultiPaX
	MultiPaX_0001D	MultiPaX
	MultiPaX_00028	MultiPaX
	MultiPaX_00029	MultiPaX

MultiPaX pipeline

## 2. Prerequisites

As MultiPaX comes in form of a GATE component, you will obviously need [GATE](#) itself. Most of the required pre-processing components are included in the GATE distribution. In particular, you will need (see the pipeline configuration details below): 1. Tokenizer, 2. Sentence Splitter, 3. POS-Tagger, 4. Morphological Analyser (optional, to get root forms of triples for Stanford Parser and MuNPEX), 5. One Parser: RASP-3, MiniPar, Stanford Parser, SUPPLE, or noun phrase (NP) extractor (e.g., [MuNPEX](#))

## 3. Documentation

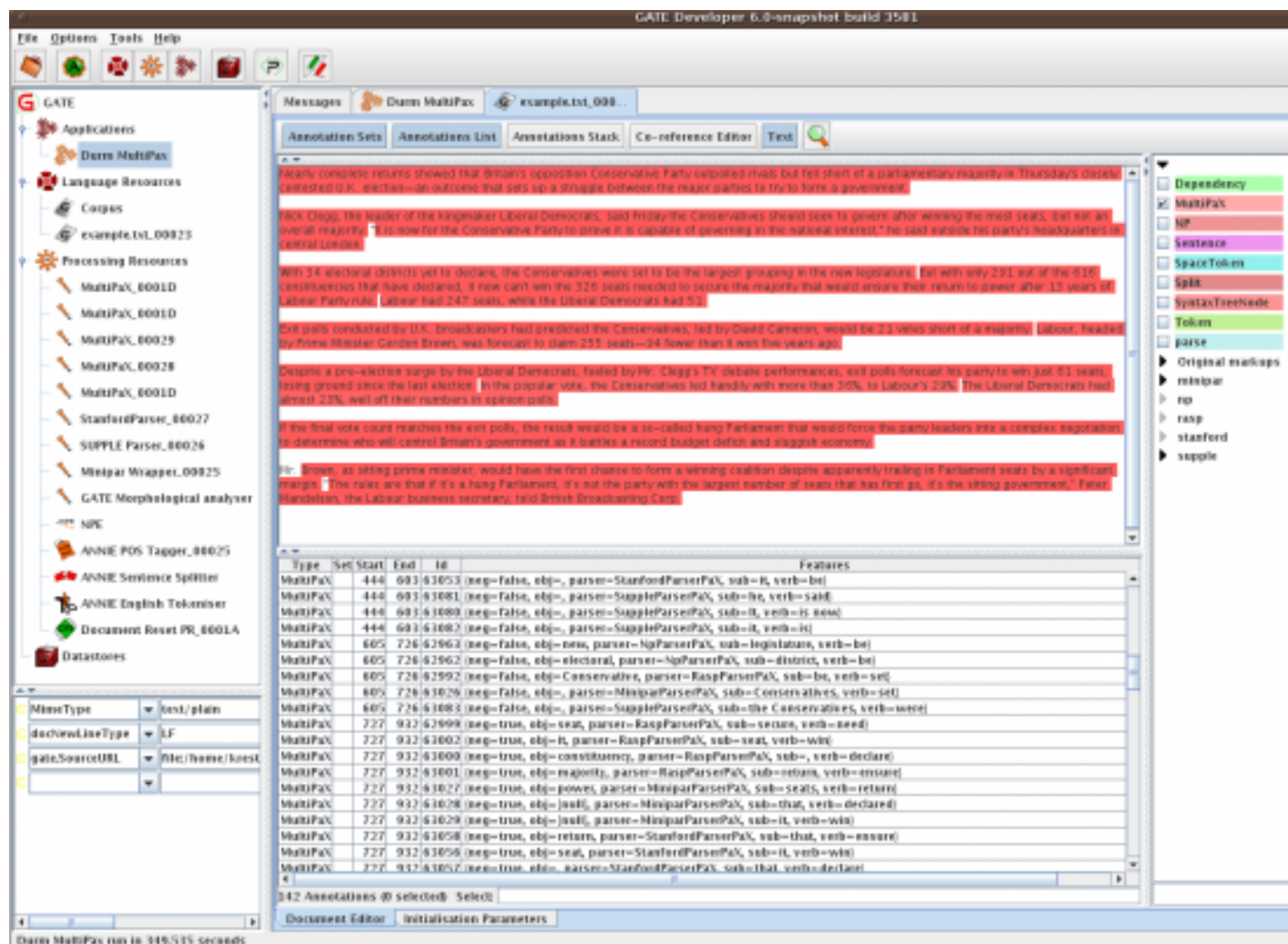
The MultiPaX component is designed to be embedded in more complex pipelines. Here, we describe the minimum requirements for obtaining predicate-argument structures annotations. Note that in the following discussion we assume you know how to work with GATE, for tasks like adding a new CREOLE repository or loading new components into a processing pipeline. If you haven't done this before, please read the [GATE user's guide](#) first!

## 4. Pipeline Configuration

The minimal pipeline to get the predicate-argument structures for all parsers is shown in the figure above. Since there is no wrapper in GATE for RASP-3 we included one in the MultiPaX component. To use RASP-3 you have to specify the location of the RASP script as a parameter when you load the MultiPaX component.

## 5. MultiPaX Configuration

The MultiPaX component has four runtime parameters. The most important one is "parserASName". It tells the MultiPaX component which parser output is used to extract the triples. Valid entries are: "rasp", "supple", "minipar", "stanford", or "np". Make sure the output annotation set name of the SUPPLE parser wrapper is set accordingly (semanticsSetName="supple"). The same holds if you want to use MiniPar (annotationSetOutputName="minipar").



Screenshot of GATE with MultiPaX annotations

## 6. Result Annotations

A new annotation set of type MultiPaX is added to the document. If the component detects a predicate-argument structure in the output of the selected parser, the sentence containing the PAS is annotated and "sub", "obj", "verb", "neg", and "parserName" properties are added to the annotation.

An example of the result annotations can be seen in the figure on the left.

## 7. Downloads

- the GATE MultiPaX component [Version 1.3](#)
- our research paper about the [Predicate-Argument-EXtractor](#)
- the GNU GPL [license](#), under which you can use the component

If you use our component, please cite our paper: [Krestel, R., R. Witte, and S. Bergler](#), "Predicate-Argument EXtractor (PAX)", *New Challenges for NLP Frameworks*, Valletta, Malta : ELRA, pp. 51--54, May 22, 2010.

## 8. Feedback

For questions, comments, etc., please use the [Forum](#).

## 9. Version history

- 1.3: 02.12.2010. OutputASName can now be set to arbitrary names.
- 1.2: 10.08.2010. Morphological Analyzer is now an optional requirement.
- 1.1: 10.05.2010. Improved LREC release.
- 1.0: 01.03.2010. Initial public release.

---

## References

1. [Krestel, R., R. Witte, and S. Bergler. "Predicate-Argument EXtractor \(PAX\)". \*New Challenges for NLP Frameworks\*. Valletta, Malta : ELRA, pp. 51--54, May 22, 2010.](#)



Except where otherwise noted, all original content on this site is copyright by its author and licensed under a [Creative Commons Attribution-Share Alike 2.5 Canada License](#).

Source URL (retrieved on 2025-12-22 13:04): <https://www.semanticsoftware.info/multipax>