

Apache UIMA™ Ruta

Rule-based Text Annotation

Version 2.1.0

<http://uima.apache.org/ruta.html>

Peter Klügl

What is UIMA Ruta?

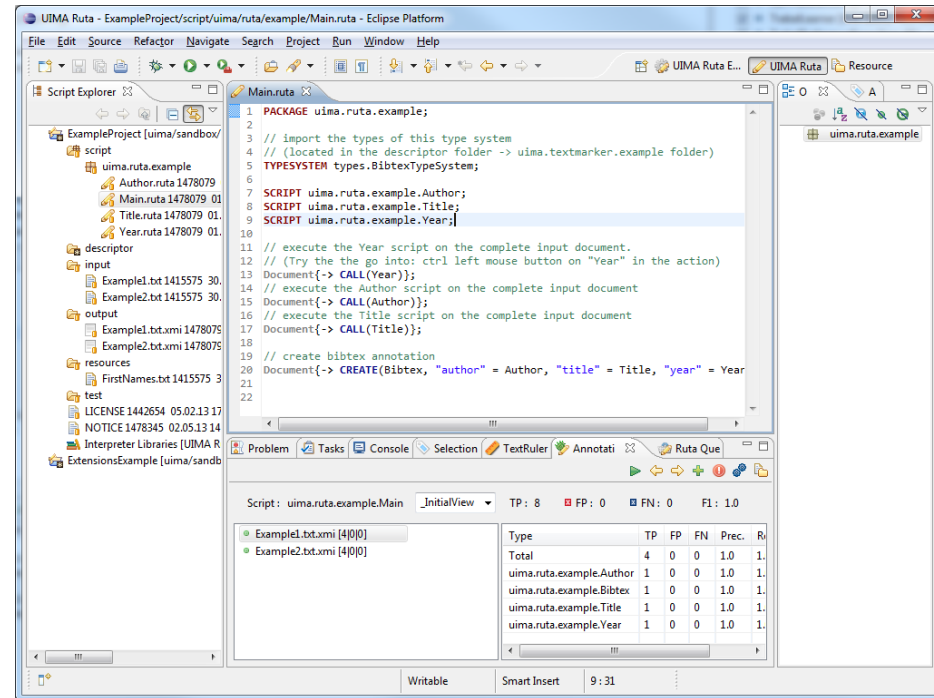
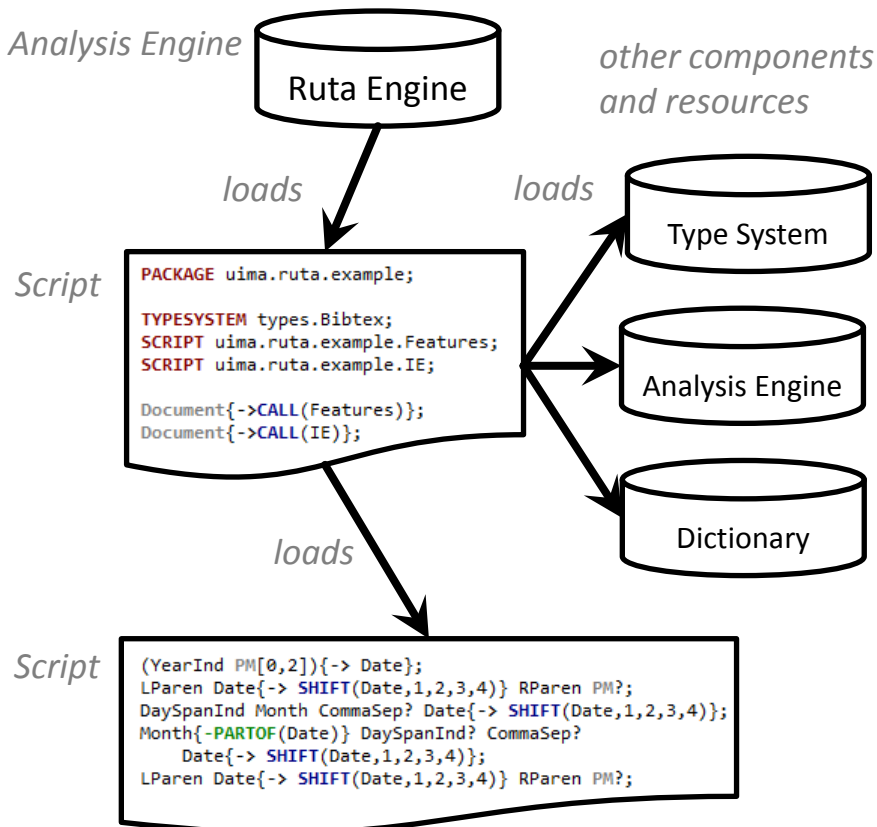
Rule-based script language interpreted by a generic Analysis Engine

2008: First TextMarker release on SourceForge
2011: TextMarker contributed to Apache UIMA
2013: Renamed to UIMA Ruta
2013: Version 2.1.0 released



Eclipse-based development environment and tooling:

UIMA Ruta Workbench

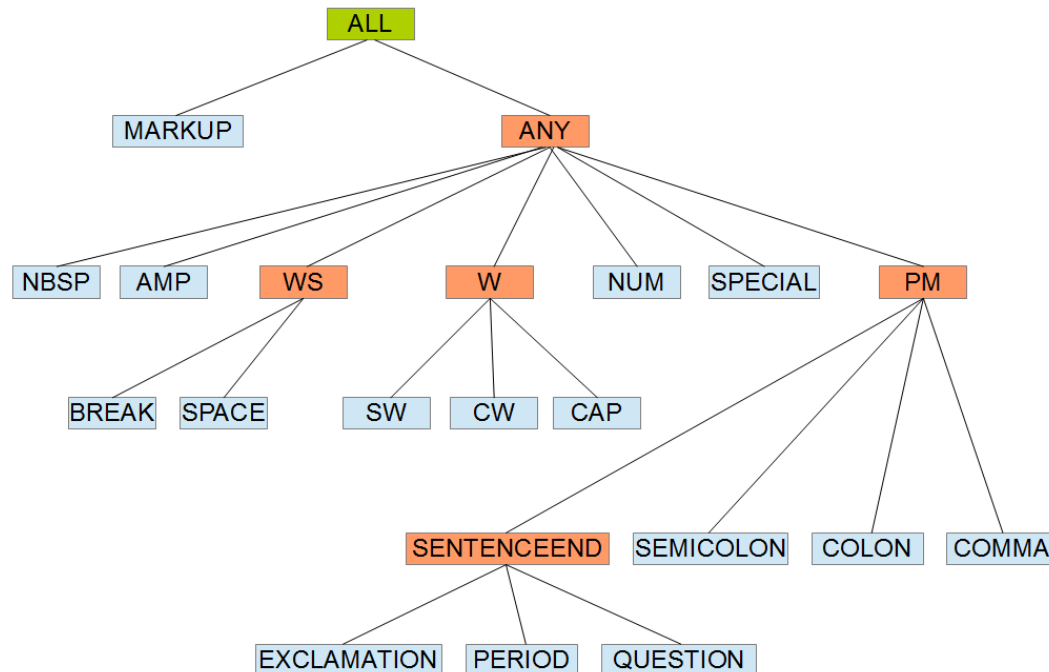


Agenda

- I. **UIMA Ruta Language**
- II. UIMA Ruta Workbench

Annotation seeding

- Provide some initial annotations
- Seeding is extensible



(Simplified) Script Syntax

Script → Package? Import* Statement*
Import → (“TYPESYSTEM” | “SCRIPT” | “ENGINE” | “UIMAFIT”)
Identifier “;”
Statement → (Declaration | Rule | Block)
Block → “BLOCK” “(” Identifier “)” RuleElement “{” Statement+ “}”

```
PACKAGE uima.ruta.example; ← Package

TYPESYSTEM types.BibtexTypeSystem;
SCRIPT uima.ruta.example.Author; ← Import
SCRIPT uima.ruta.example.Title;
SCRIPT uima.ruta.example.Year;

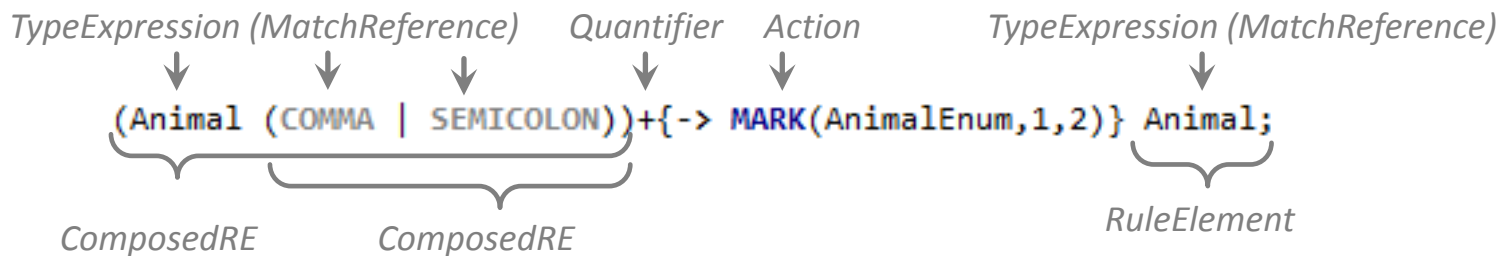
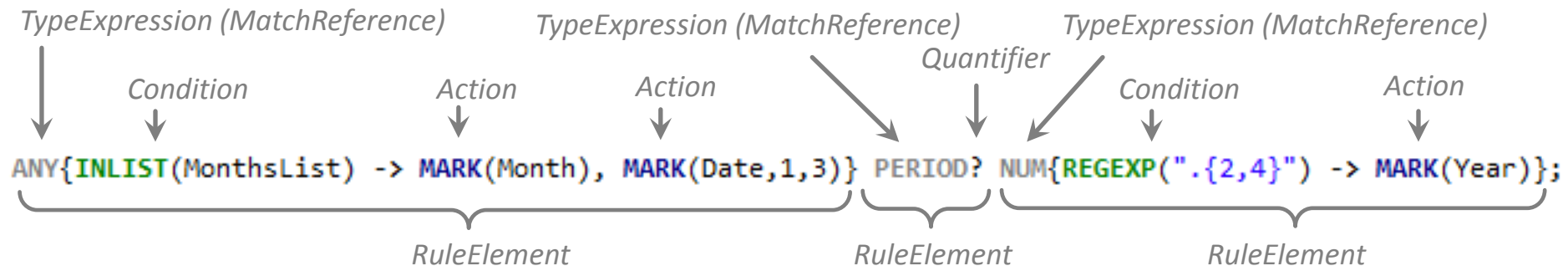
DECLARE Reference; ← Declaration
WORDLIST FirstNameList = 'FirstNames.txt';

Document{-> MARKFAST(FirstName, FirstNameList)};
Document{-> CALL(Year)};
Document{-> CALL(Author)}; ← Rule
Document{-> CALL(Title)};

BLOCK(forEach) Reference[]{ ← Block
    Document{-> CREATE(Bibtex, "author" = Author,
        "title" = Title, "year" = Year)};
}
```

(Simplified) Rule Syntax

Rule → (RuleElement+ | RegExpRule | ConjunctRules) ";"
 RuleElement → MatchReference Quantifier? ("{" **Conditions**?
 "->" **Actions**? "}")? InlinedRules?
 MatchReference → (TypeExpression | StringExpression | ComposedRE | WildCard)
 ComposedRE → "(" RuleElement ((" & " | "|")? RuleElement)* ")"



Rule Syntax: Syntactic Sugar

- **MatchReference:** FeatureExpression, FeatureMatchExpression

```
Dependency.governor CW;      Token.pos.begin == 0;
```

- **Implicit Actions:** TypeExpression, FeatureAssignmentExpression

```
Paragraph{CONTAINS(B) -> Headline}; Paragraph{ -> Paragraph.begin = 0};
```

- **Implicit Conditions:** BooleanExpression, FeatureMatchExpression

```
CW{CW.begin > 10};          CW{boolVar -> MARK(SomeType)};
```

```
ANY{INLIST(MonthsList) -> MARK(Month), MARK(Date,1,3)} PERIOD? NUM{REGEXP(".{2,4}") -> MARK(Year)};
```



TypeExpression instead of MARK action

```
(ANY{INLIST(MonthsList) -> Month} PERIOD? NUM{REGEXP(".{2,4}") -> Year}){ -> Date};
```

Rule Inference

Basic algorithm:

1. Find valid positions for first rule element
 2. Evaluate if following rule element can match next to previous position (repeat for all rule elements)
 3. Apply actions if complete rule successfully matched
- Composed rule elements delegate to their inner elements
 - Quantifiers specify how often rule element matches:

? ?? * *? + +? [1,2] [1,2]?

```
ANY{INLIST(MonthsList) -> MARK(Month), MARK(Date,1,3)} PERIOD? NUM{REGEXP(".{2,4}") -> MARK(Year)};
```

Dec. 2004, July 85, 11.2008 → *Dec. 2004*, *July 85*, *11.2008*

| |
|-------|
| Date |
| Month |
| Year |

Rule Inference

- Imperative rule execution
- Based on complete disjoint partitioning (RutaBasic)
- Depth-first matching
 - Complete current alternative before matching the next one

```
ANY+{-PARTOF(Text) -> Text};          PERIOD Annotation PERIOD;
```

- Only permutations in matching direction
- Manual selection of starting rule element

```
ANY LastToken;                          ANY @LastToken;
```

- Dynamic anchoring: Guess best rule element
- Special rule element: „do not care“ wildcard

```
PERIOD ANY+?{-> Sentence} PERIOD;      PERIOD #-{-> Sentence} PERIOD;
```

Beyond Sequential Patterns

- Conjunctive rule elements

- All rule elements need to match at same position
- Use largest match to continue

```
(Token.posTag=="DET" & Lemma.value=="the");  
NUM (W{REGEXP("Peter") -> Name} & (ANY CW{PARTOF(Name)}));
```

- Disjunctive rule elements

- One rule element needs to match

```
(Animal ((COMMA | "and") Animal)+){-> AnimalEnum};  
(("Peter" CW) | ("Mr" PERIOD CW)){-> Name};
```

- Conjunct rules

- Both rules need to match anywhere in window

```
CW NUM % SW NUM{-> MARK(Found, 1, 2)};
```

Imports, Declarations and Expressions

- Supported imports
 - Scripts `SCRIPT uima.ruta.example.Author;`
 - Type Systems `TYPESYSTEM utils.PlainTextTypeSystem;`
 - Analysis Engines `ENGINE utils.PlainTextAnnotator;`
 - uimaFIT Analysis Engines
`UIMAFIT de.tudarmstadt.ukp.dkpro.core.tokit.BreakIteratorSegmenter;`
- Supported declarations:
 - Types `DECLARE SimpleType1, SimpleType2;`
`DECLARE ParentType NewType (SomeType feature1, INT feature2);`
 - Variables for types, int, strings, booleans, lists, ...
 - Resources `WORDLIST listName = 'someWordList.txt';`
`WORDTABLE tableName = 'someTable.csv';`
- Supported Expressions
 - Primitive types, variables, functions
 - String concatenations, boolean comparison, operations on numbers, ...
- All arguments of conditions/actions are expressions

Actions, Conditions and Functions

Language provides right now:

- 41 Actions
 - **MARK, UNMARK, CREATE, TRIE, TRIM, SHIFT, CALL, EXEC, ...**
- 27 Conditions
 - **CONTAINS, PARTOF, REGEXP, STARTSWITH, NEAR, ...**
- Functions for Type, Boolean, String and Number
- Extensible language definition: Add your own elements

Filtering and Visibility

- Complete document is modelled
- No restriction to tokens
- Invisible types = (default + filtered) – retained
- Annotations are invisible, if their begin or end is invisible
- Filtering adaptable by rules
- Make uninteresting parts invisible (default: space, markup, ...)

```
Sentence;  
Document{-> RETAINTYPE(SPACE)};  
Sentence;  
Document{-> FILTERTYPE(CW)};  
Sentence;  
Document{-> RETAINTYPE, FILTERTYPE};
```

```
W NUM; matches on → Dec<br>2004, July 85, May1999  
Document{-> RETAINTYPE(SPACE, MARKUP)};  
W NUM; matches on → May1999
```

Blocks and Inlined Rules

- BLOCK construct

- Modularize scripts beyond files
- Conditioned statements
- Windowing
- Foreach loops
- Procedures (recursion)

```
BLOCK(German) Document{FEATURE("language", "de")} {  
    // rules for german documents  
}  
BLOCK(ForEach) Sentence{-STARTSWITH(NP)} {  
    // ... do something  
}
```

- Inlined Rules

- As „actions“: ->
 - Simplified block constructs
- As „conditions“: <-
 - Nested conditions

```
Sentence->{  
    Document{-STARTSWITH(NP) -> SentNoLeadingNP};  
};  
  
Sentence{-> SentenceWithNPNP}<-{  
    NP NP;  
};
```

Scoring Rules

- ... for dealing a bit with uncertainty
- ... for weighting different aspects
- Action **MARKSCORE** adds score
- Condition **SCORE** validates score against a threshold

```
STRING s;  
Paragraph{CONTAINS(W,1,5) -> MARKSCORE(5,HeadlineInd)};  
Paragraph{CONTAINS(W,6,10) -> MARKSCORE(2,HeadlineInd)};  
Paragraph{CONTAINS(Bold,80,100,true) -> MARKSCORE(7,HeadlineInd)};  
Paragraph{CONTAINS(Bold,30,80,true) -> MARKSCORE(3,HeadlineInd)};  
Paragraph{CONTAINS(CW,50,100,true) -> MARKSCORE(7,HeadlineInd)};  
Paragraph{CONTAINS(W,0,0) -> MARKSCORE(-50,HeadlineInd)};  
HeadlineInd{SCORE(10) -> MARK(Headline)};  
HeadlineInd{SCORE(5,10) -> MATCHEDTEXT(s), LOG("Maybe a headline: " + s)};
```

Simple RegExp Rules

```
RegExpRule      → StringExpression "->" GroupAssignment  
                ("," GroupAssignment)* ";"  
GroupAssignment → TypeExpression FeatureAssignment?  
                | NumberExpression "=" TypeExpression FeatureAssignment?  
FeatureAssignment → "(" StringExpression "=" Expression  
                  ("," StringExpression "=" Expression)* ")"
```

- Match on regular expressions (supports capturing groups)
- No restrictions due to partitioning or visibility

```
DECLARE T1, T2;
```

```
DECLARE Annotation Complex (STRING s, Annotation a, BOOLEAN b);
```

```
"A(.*)C" -> T1, 1 = T2;
```

```
"B(.*)B(.)" -> 1 = Complex ("s" = 0, "a" = 2, "b" = true),  
              2 = Complex ("s" = 0, "a" = 1, "b" = false);
```


Analysis Engines and Type systems

- UIMA Ruta
 - BasicEngine.xml (RutaEngine.class) (includes TypePriorities.xml)
 - BasicTypeSystem.xml (includes InternalTypeSystem.xml)
- Additional Analysis Engines shipped with UIMA Ruta
 - AnnotationWriter
 - Cutter
 - HtmlAnnotator (with Type System)
 - HtmlConverter
 - Modifier
 - PlainTextAnnotator (with Type System)
 - ViewWriter
 - XMIWriter

Configuration Parameters

▼ Configuration Parameters

This section lists all configuration parameters, either as plain parameters, or as part of one or more groups. Select one to show, or set the value in the right hand panel.

| | |
|----------------------|--------------------------------|
| ▲ <Not in any group> | |
| Multi Opt String | Name: seeders |
| Single Opt Boolean | Name: debug |
| Multi Opt String | Name: additionalScripts |
| Single Opt Boolean | Name: profile |
| Single Opt Boolean | Name: debugWithMatches |
| Single Opt Boolean | Name: statistics |
| Multi Opt String | Name: additionalEngines |
| Multi Opt String | Name: additionalExtensions |
| Multi Opt String | Name: debugOnlyFor |
| Single Opt String | Name: scriptEncoding |
| Multi Opt String | Name: additionalEngineLoaders |
| Multi Opt String | Name: resourcePaths |
| Multi Opt String | Name: defaultFilteredTypes |
| Single Opt String | Name: mainScript |
| Multi Opt String | Name: scriptPaths |
| Multi Opt String | Name: descriptorPaths |
| Single Opt Boolean | Name: removeBasics |
| Single Opt Boolean | Name: dynamicAnchoring |
| Single Opt Boolean | Name: lowMemoryProfile |
| Single Opt Boolean | Name: createdBy |
| Single Opt Boolean | Name: simpleGreedyForComposed |
| Multi Opt String | Name: additionalUimafitEngines |

Control of Analysis Engine

- mainScript
- scriptPaths
- descriptorPaths
- additionalScripts
- ...

Explanation of Analysis Engine

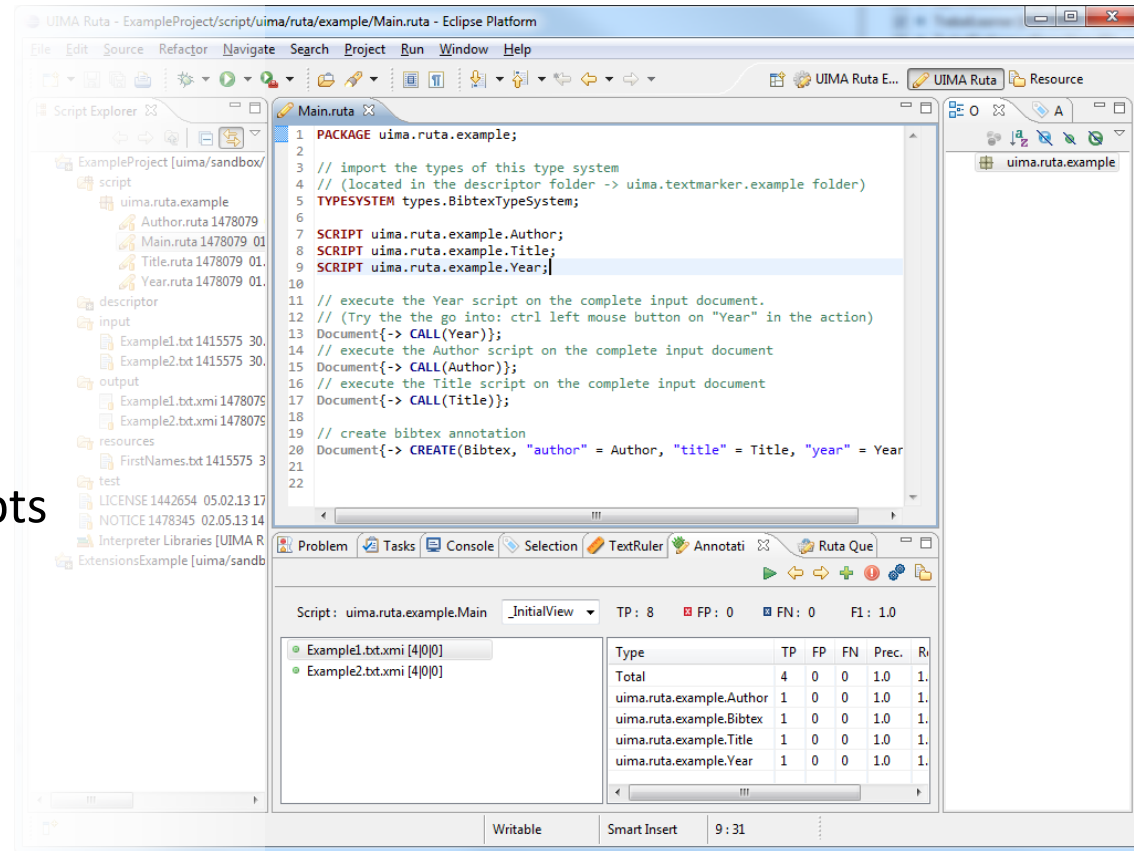
- debug
- profile
- ...

Agenda

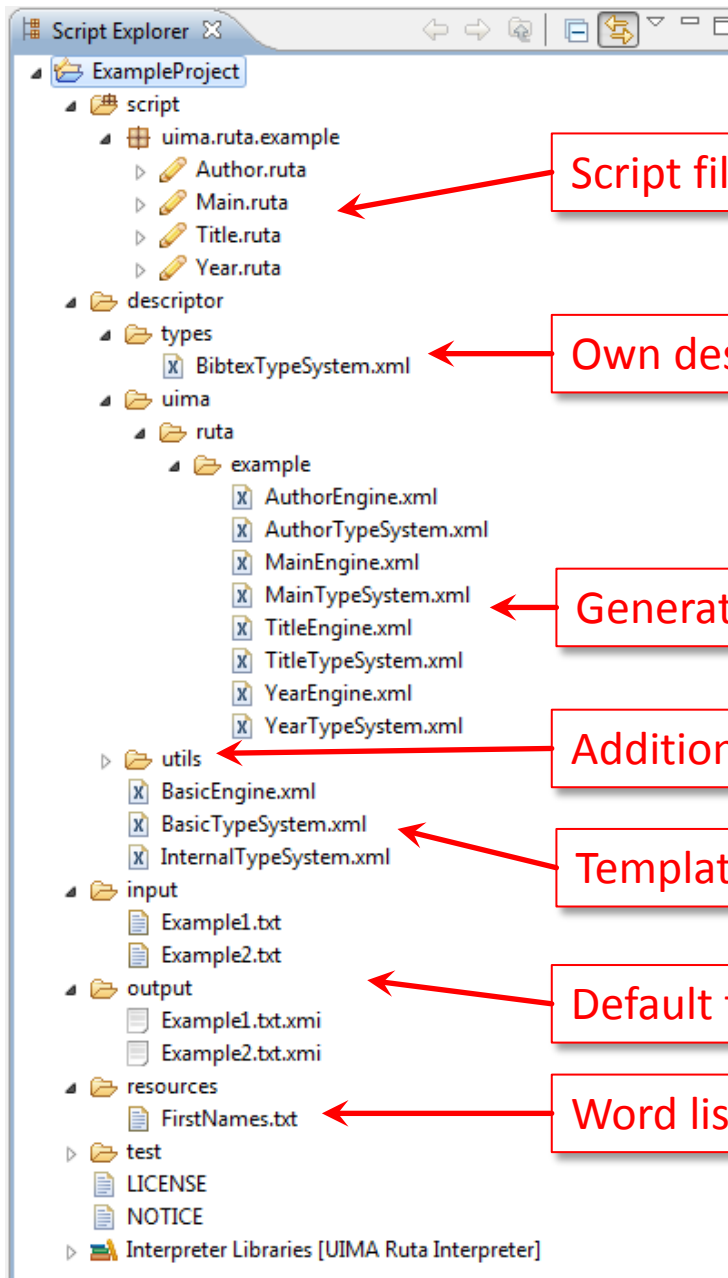
- I. UIMA Ruta Language
- II. UIMA Ruta Workbench**

UIMA Ruta Workbench: IDE for UIMA Ruta rules

- Full-featured rule editor
 - Syntax highlighting
 - Semantic highlighting
 - Syntax checking
 - Auto-completion
 - Template-based completion
 - Open declaration
 - Formatter
- Generates descriptors for scripts
 - Analysis Engine
 - Type System
- Many useful tools
- Supports Mixin-Workspaces
 - Dependencies to Java projects



UIMA Ruta Project Layout



Script files

Own descriptors

Generated descriptors

Additional components

Template descriptors

Default folders for launch config

Word lists

UIMA Ruta Explain Perspective

The screenshot displays the Apache UIMA Ruta Explain Perspective interface. It is divided into several panes:

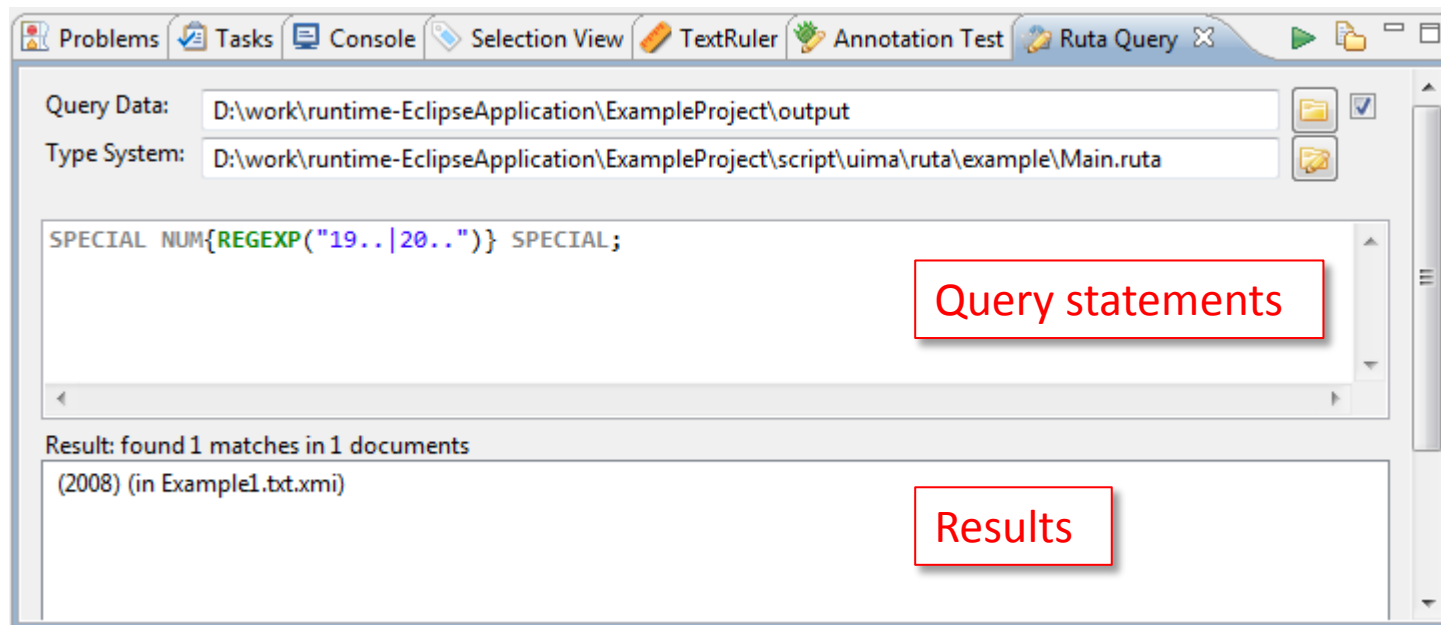
- Left Pane:** Contains filters for 'Only types with...' and 'Only annotations with...'. A list of annotations is shown, including 'BA d.u.e.Bibtex [1]', 'BA d.u.e.Title [1]', 'BA d.u.e.T.TitleStopper [1]', 'BA d.u.e.Year [1]' (with '(2008)' selected), 'BA o.a.u.t.t.BREAK [2]', 'BA o.a.u.t.t.CAP [5]', and 'BA o.a.u.t.t.COLON [3]'. Below this is the 'Rule Elements' pane, showing the rule: `SPECIAL{REGEXP("[()]} Year{ -> UNMARK(Year),MARK(Year,1, 2, 3)} SPECIAL{REGEXP("[()]} Year{ -> UNMARK(Year),MARK(Year,1, 2, 3)} SPECIAL{REGEXP("[()]} Year{ -> UNMARK(Year),MARK(Year,1, 2, 3)}`.
- Top-Right Pane:** Shows the 'Applied Rules' tree. The selected rule is `[1/1] BLOCK(Main) Document [0.094s]`. It lists several sub-rules with their application counts and success rates, such as `[1/1] Document{ -> CALL(Year)}` and `[1/1] BLOCK(Year) Document`.
- Bottom-Left Pane:** Shows 'Failed Rules' (Bethard, S.) and 'Matched Rules' (Ogren, P.V., Wetzler, P.G.).
- Bottom-Right Pane:** Shows 'Rule Elements' for a specific rule application. It lists elements like 'Name{-PARTOF(NameListPart)}' (Bethard, S.), 'Name' (checked), '-PARTOF(NameListPart)' (checked), 'PARTOF(NameListPart)' (failed), 'NameLinker[1,2]{ -> MARK(NameListPart, 1, 2)}' (checked), and 'NameLinker' (failed).

Red text boxes and arrows highlight key areas:

- Top-Left:** A red box asks "What rule created this annotation?" with an arrow pointing to the 'BA d.u.e.Year [1]' annotation.
- Top-Right:** A red box asks "Complete listing of rule applies:" followed by a list: "How often tried", "How often succeeded", and "Profiling information". An arrow points to the 'Applied Rules' tree.
- Bottom-Left:** A red box asks "Where did this rule match or fail?" with an arrow pointing to the 'Matched Rules' list.
- Bottom-Right:** A red box asks "Why did this rule (not) match on this position?" with an arrow pointing to the 'Rule Elements' pane.

Ruta Query View

- Use rules as query statements
- View displays rule matches in a set of documents



Annotation Testing View

- Unit tests for UIMA Ruta scripts
- Compare result of rules against gold documents
- True Positives, False Positives and False Negatives are displayed in CAS Editor
- Different engineering approaches
 - Create gold document → test-driven development of rules
 - Store correct result of rules as test → backtesting when rules are extended/modified

Script: uima.ruta.example.Main _InitialView TP : 8 FP : 0 FN : 0 F1 : 1.0

Example1.txt.xmi [4|0|0]
Example2.txt.xmi [4|0|0]

Gold documents

| Type | TP | FP | FN | Prec. | Recall | F1 |
|--------------------------|----|----|----|-------|--------|-----|
| Total | 4 | 0 | 0 | 1.0 | 1.0 | 1.0 |
| uima.ruta.example.Author | 1 | 0 | 0 | 1.0 | 1.0 | 1.0 |
| uima.ruta.example.Bibtex | 1 | 0 | 0 | 1.0 | 1.0 | 1.0 |
| uima.ruta.example.Title | 1 | 0 | 0 | 1.0 | 1.0 | 1.0 |
| uima.ruta.example.Year | 1 | 0 | 0 | 1.0 | 1.0 | 1.0 |

Evaluation results

Constraint-driven Evaluation (CDE)

- Formalize expectations on domain as constraints:
 - with UIMA Ruta rules
 - with Annotation Distributions
- Test rules (the output) on expectations
- No labeled data needed
- ... but can be used to develop constraints
- Greatly improves rule engineering

The screenshot displays the Eclipse Platform interface for UIMA Ruta CDE. The main editor shows a document with text extracted from a PDF, with various parts highlighted in different colors (green, purple, red, yellow). The interface includes several panels:

- Documents:** A panel on the right showing the list of documents being processed. It includes a table with columns for Document, CDE, and F1 scores.
- Prediction:** A panel on the right showing the results of the CDE evaluation for each document, including CDE and F1 scores.
- Define constraints:** A panel at the bottom left showing a list of constraints and their weights.
- Results of constraints:** A panel at the bottom right showing the results of the constraints for each document, including the constraint name and the resulting score.

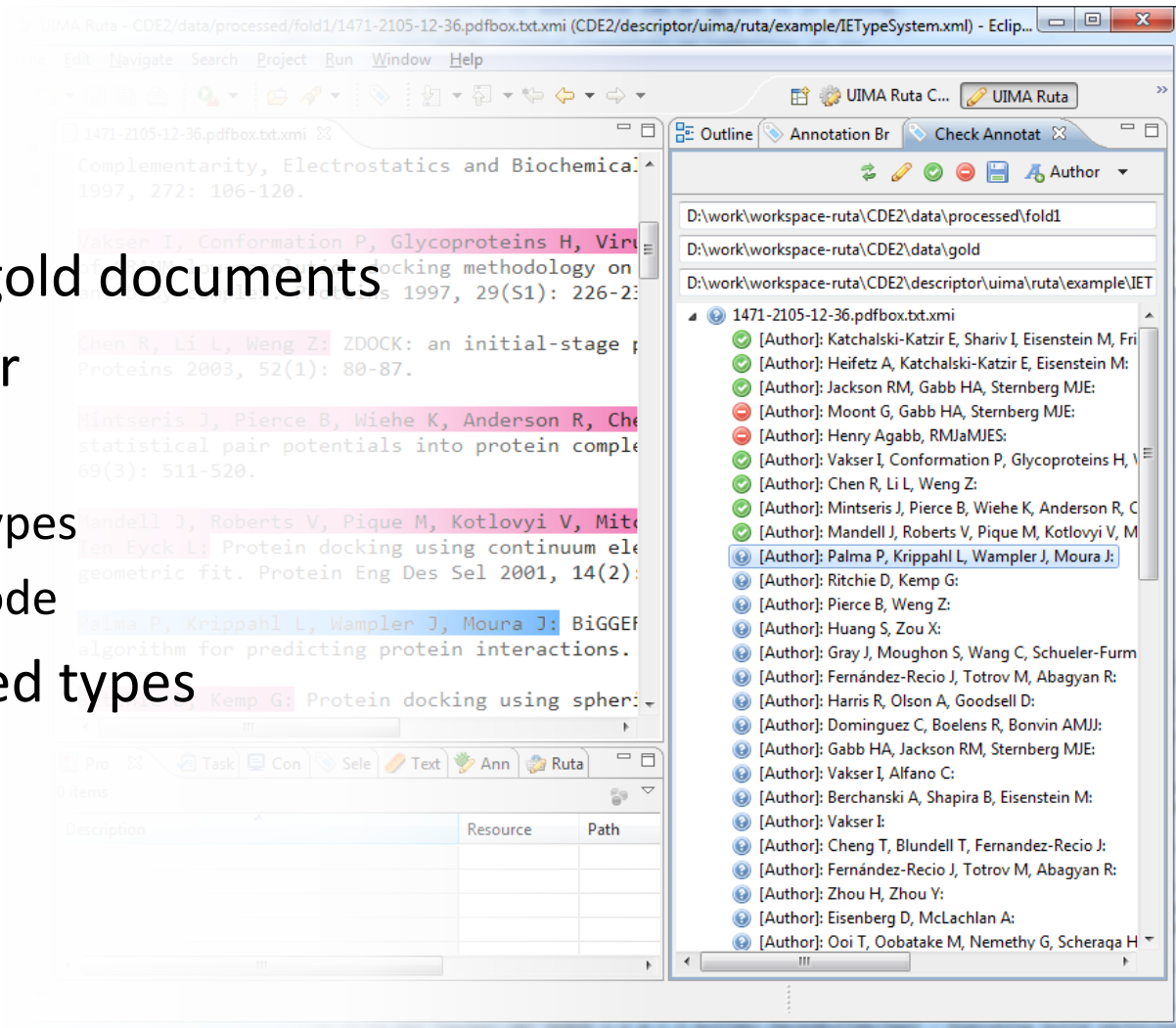
| Document | CDE | F1 |
|--------------------------------|--------|--------|
| kdm112.pdfbox.txt.xmi | 0.952 | 0.8936 |
| A97-1010.txt.xmi | 0.958 | 0.9371 |
| mldm_2_2_80-99.pdfbox.txt.xmi | 0.9657 | 0.9444 |
| A00-2002.txt.xmi | 0.978 | 0.9474 |
| A88-1009.txt.xmi | 0.987 | 0.9636 |
| A94-1026.txt.xmi | 0.9881 | 1.0 |
| J05-4002.txt.xmi | | |
| C02-1020.txt.xmi | | |
| J05-2005.txt.xmi | | |
| mldm_2_1_3-22.pdfbox.txt.xmi | 0.994 | 0.9782 |
| 1471-2105-12-36.pdfbox.txt.xmi | 0.9947 | 0.9923 |
| J05-1003.txt.xmi | 0.9947 | 0.9875 |
| 1471-2105-12-43.pdfbox.txt.xmi | 0.997 | 0.9934 |
| 1471-2105-12-37.pdfbox.txt.xmi | 1.0 | 1.0 |
| A00-1042.txt.xmi | 1.0 | 1.0 |
| C02-1035.txt.xmi | 1.0 | 1.0 |
| C04-1034.txt.xmi | 1.0 | 1.0 |

| Constraint | Weight |
|--|--------|
| Reference(OR(STARTSWITH(Author), STARTSWITH(Editor))); | 1 |
| Author(-CONTAINS(NUM)); | 1 |
| Author(Date Title); | 1 |
| Author(CONTAINS(CW,1,100)); | 1 |
| Author(CONTAINS(W,2,200)); | 1 |
| Author(-CONTAINS(EditorMarker)); | 1 |
| Author(STARTSWITH(Reference)); | 1 |

| Constraint | Result |
|--|--------------------|
| Reference(OR(STARTSWITH(Author), STARTSWITH(Editor))); | 0.8461538461538461 |
| Author(-CONTAINS(NUM)); | 1.0 |
| Author(Date Title); | 0.9090909090909090 |
| Author(CONTAINS(CW,1,100)); | 1.0 |
| Author(CONTAINS(W,2,200)); | 0.9090909090909090 |
| Author(-CONTAINS(EditorMarker)); | 1.0 |
| Author(STARTSWITH(Reference)); | 1.0 |

Check Annotations view

- Support creation of gold documents
- Control for CAS Editor
 - Specify type system
 - Specify highlighted types
 - Select annotation mode
- Remembers processed types



TextRuler: Framework for rule learning algorithms

The screenshot displays the Eclipse IDE interface for UIMA Ruta. The main editor shows the `Features.ruta` script, which defines the package, engine, typesystem, and wordlists. The `TextRuler` configuration window is open, showing training and test data paths, preprocess scripts, and selected information and filtered feature types. The `Methods` section lists available algorithms like `KEP`, `LP2`, `TraBaL`, and `WHISK`. The `Outl` view on the right shows the generated Ruta script, which includes rules for tokenization and block detection.

Gold data and features

Target types

Available algorithms

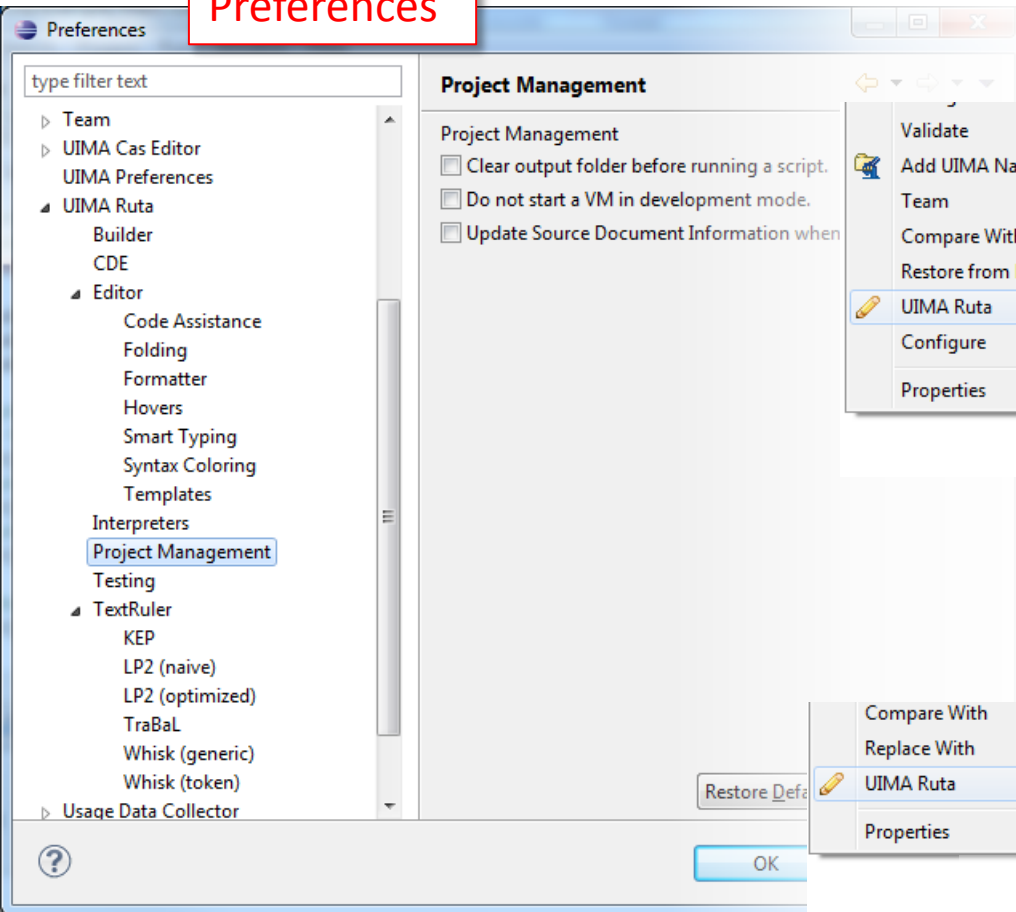
Result

TextRuler: Available algorithms

- **LP² [Ciravegna, 2003]**
 - Boundary matching rules
 - Context rules for filling gaps in boundary rules
 - No correction rules (yet)
- **Whisk [Soderland et al., 1999]**
 - Rules in the form of modified regular expressions
 - No multi-slot rules (yet)
- **KEP**
 - Basic idea: How does a human write rules?
 - Set of simple algorithms for different engineering patterns
 - Exploit synergy by combination
- **Trabal**
 - Transformation-based rule learner
 - Try to learn rules that correct existing annotations

Preferences and Popup Commands

Preferences



Apply script on folder

Update UIMA Ruta Project

Compile word lists

Summary

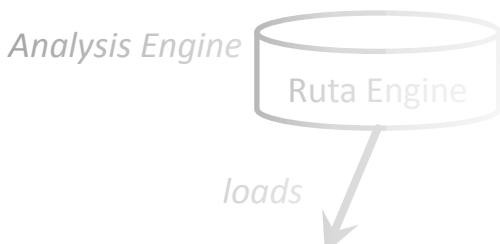
Rule-based script language interpreted by a generic Analysis Engine

Eclipse-based development environment and tooling:

Apache UIMA Ruta

Rule-based Text Annotation

<http://uima.apache.org/ruta.html>



```
Script
PACKAGE uima.ruta.example;

TYPESYSTEM types.Bibtex;
SCRIPT uima.ruta.example.Features;
SCRIPT uima.ruta.example.IE;

Document{->CALL(Features)};
Document{->CALL(IE)};
```

```
Script
(YearInd PM[0,2]){-> Date};
LParen Date{-> SHIFT(Date,1,2,3,4)} RParen
DaySpanInd Month CommaSep? Date{-> SHIFT(Date,1,2,3,4)}
Month{-PARTOF(Date)} DaySpanInd? CommaSep?
Date{-> SHIFT(Date,1,2,3,4)};
LParen Date{-> SHIFT(Date,1,2,3,4)} RParen
```

- Compact and powerful language for text processing tasks
- Supports different approaches
- Designed for rapid development
- Extensible and combinable
- Serious engineering support

