UIMA & U-COMPARE

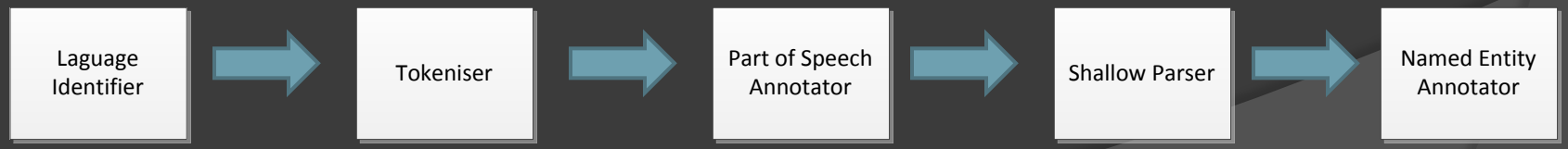
METANET4U

Overview

- Introduction
- Overview of UIMA & U-Compare Frameworks
- Aim of work
  - Extending & creating compatible Components
- Demonstration

Introduction

- METANET4U
- Natural Language Processing
  - Combination of Computer Science & Linguistics
- Workflow of components
  - Example: Named Entity Recognition



UIMA

- Unstructured Information Management Architecture (UIMA)
- Unstructured Information
- Analysing unstructured content
- Cumbersome to connect arbitrary tools

UIMA Architecture Overview

- Analysis Engines – UIMA's basic building blocks
- Analysis Results
  - E.g.: *The span from position 101 to 112 in document D102 denotes a Person*
- Annotators – holding the core analysis algorithms

UIMA Architecture Overview

- Common Analysis Structure (CAS) – object-based data structure
- Type system – an object schema for the CAS
  - Features – defining properties of a Type
- Annotation Type – general Type used in artefact analysis
  - Being & end features

UIMA Solution

- Developing a UIMA solution:
  - Define the CAS types that the Annotator will use
  - Generate Java classes for the CAS types
  - Write the actual Annotator Java code
  - Create the Analysis Engine descriptor
  - Test the Analysis Engine

Enhancements

- What is this architecture missing?
  - Compatible type system
  - Basic NLP functionalities
  - User Interfaces for humans

U-Compare

- Joint project between:
  - University of Tokyo,
  - University of Colorado Health Science Centre, and
  - University of Manchester
- Largest collection of ready-to-use UIMA components
- Easy to use Graphical User Interface

U-Compare Features

- Drag-and-drop components to build workflows
- Comparison of components and workflows
- Exportation of workflows

U-Compare Component

- U-Compare component compatibility
- Common type system to achieve interoperability
- UIMA component with U-Compare Type System
  - Types rather than String fields (e.g. POS token)

U-Compare Component

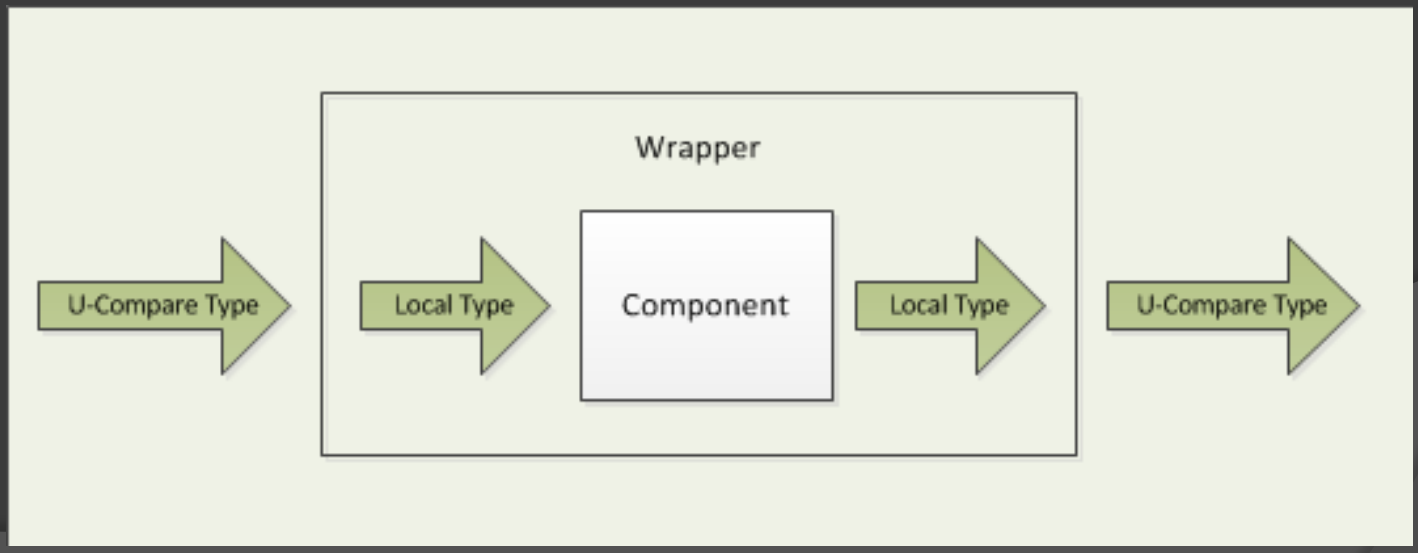
- Developing a U-Compare Component
  - The Annotator would make use of the U-Compare type system
  - The Java classes for the CAS types are provided by U-Compare
  - Write the actual Annotator Java code
  - Create the Analysis Engine descriptor
  - Test the Analysis Engine

Our tools for U-COMPARE

- Splitters
  - Paragraph Splitter
  - Sentence Splitter
- Tokeniser
- Part of Speech Tagger for Maltese

Adapting Existing Tools

- Extracting core functionality
- Wrapping existing tools to be compliant with U-Compare Type System



Paragraph Splitter Annotator Example

- Simple example – extracting core functionality
- Using U-Compare Type System
- Setting Annotation Features
- Adding to the CAS

Paragraph Splitter Code

```
public void process(JCas aJCas) throws
AnalysisEngineProcessException {
    String txt = aJCas.getDocumentText();

    String regex = "(^.*\\S+.*$)+";

    Pattern paragraph = Pattern.compile(regex, Pattern.MULTILINE);
    Matcher matcher = paragraph.matcher(txt);
    int pos = 0;
    while(matcher.find(pos)){
        org.u_compare.shared.document.text.Paragraph paragraphAnnotation =
new org.u_compare.shared.document.text.Paragraph(aJCas);
        paragraphAnnotation.setBegin(matcher.start());
        paragraphAnnotation.setEnd(matcher.end());

        paragraphAnnotation.addToIndexes();
        pos = matcher.end();
    }
}
```

Demonstration

Questions

Thank you