ORACLE

Leveraging UIMA in Spark

Philip Ogren
Consulting Member of Technical Staff
Spark Summit, June 30, 2014

Spark + UIMA

- **U**nstructured **I**nformation **M**anagement **A**rchitecture
  - Apache project, version 2.6.0, 10+ years
  - Claim to Fame: IBM Watson
  - Text processing platform
- Spark
  - Big Data Platform
  - Not a text processing platform
- Goal
  - myRdd.flatMap(text => analyzeText(text))
  - myRdd.map(text => classifyText(text))

UIMA

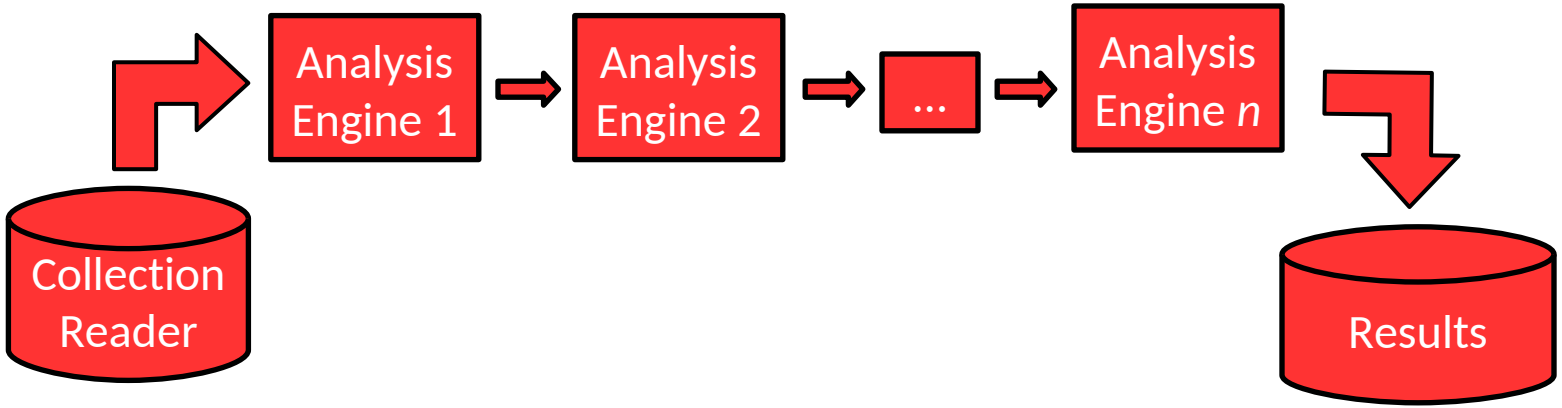
- Language Identification
- Sentence Boundary Detection
- Tokenization
- Part-of-speech tagging
- Named entity recognition
- Relation extraction
- Syntactic parsing
- Document classification
- Sentiment analysis
- ...

- Newswire
- Legal
- Medical
- Science/Bio
- Social Media
- ...

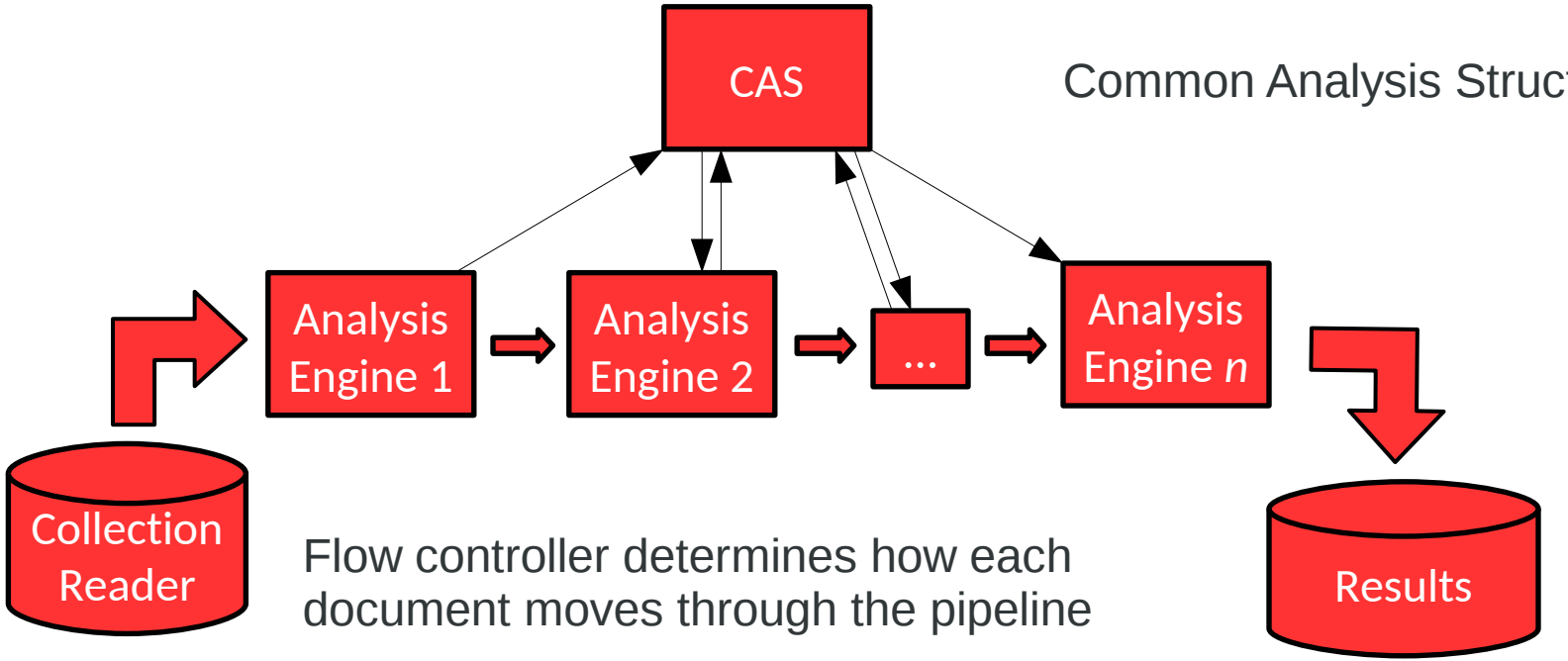
UIMA

- Framework
  - APIs
  - Configuration
  - Tooling/Infrastructure
  - Orchestration
  - Execution
- Eco-system of toolkits
  - OpenNLP
  - cTAKES
  - DKPro
  - JULIE Lab
  - BioNLP
  - ClearTK
  - UIMA Addons and Sandbox

UIMA



UIMA



CAS

- **C**ommon **A**nalysis **S**tructure
  - One per “document”
- ***content*** and ***analysis*** of document
  - Multiple views
    - Native format => plain text
    - Gold standard, system results
  - Annotations
    - Tokens, sentences, paragraphs
    - Syntactic structure, classifications, named entities

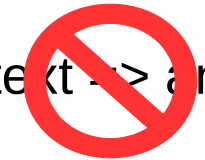
CAS

- type system
  - Expressive and flexible representation
  - Define annotations
    - (e.g. token, sentence, part-of-speech, author, title, address, email, etc.)
- indexing
  - getBookTitle(author, BEFORE)
  - for(Sentence sentence : getSentences())
    for(Token token : getTokens(sentence)) { //do something on each token }
  - List<Feature> features = extractor.extract(CAS, token));
- High performance, efficient data structure

Configuration

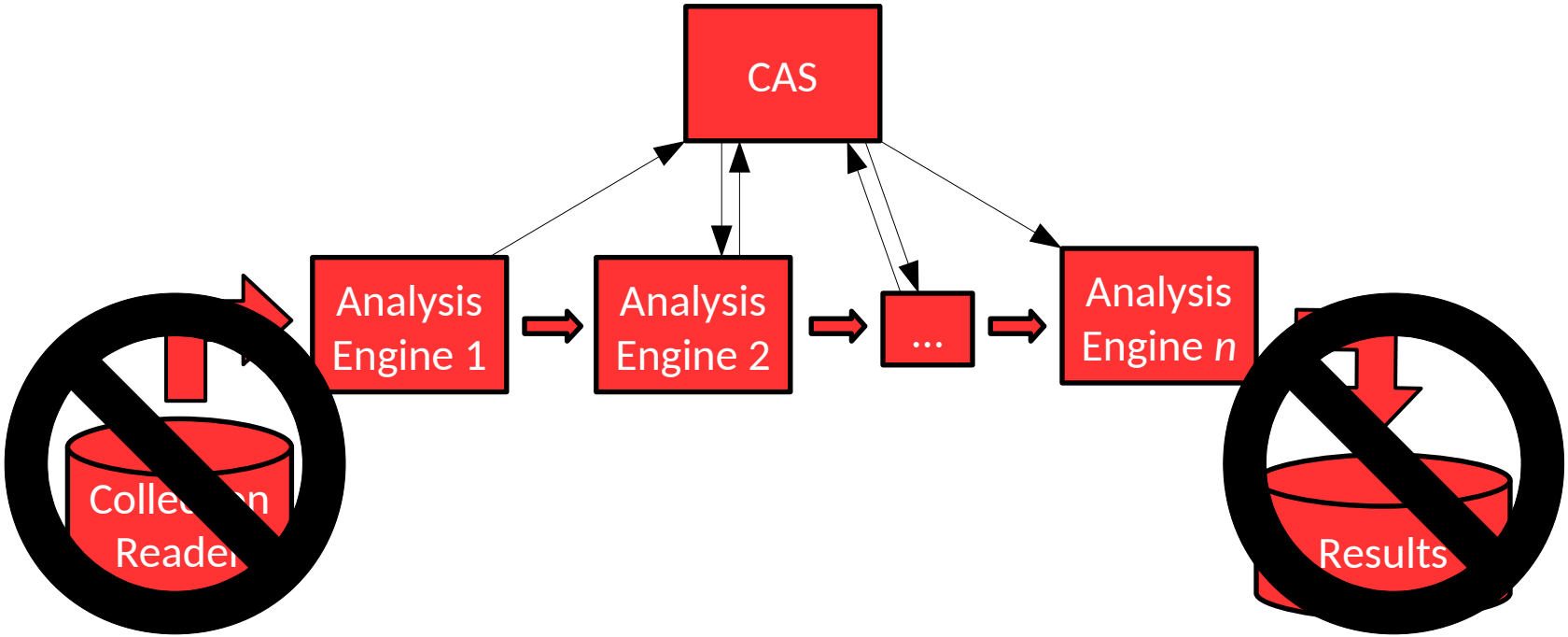
- wrap your tagger as a UIMA analysis engine
- write a descriptor file for your analysis engine
- write a CAS consumer that produces the desired output
- write another descriptor file for the CAS consumer
- write a descriptor file for a collection reader
- write a descriptor file that describes a pipeline
- invoke the Collection Processing Manager with your pipeline descriptor file

myRdd.flatMap(text -> analyzeText(text))

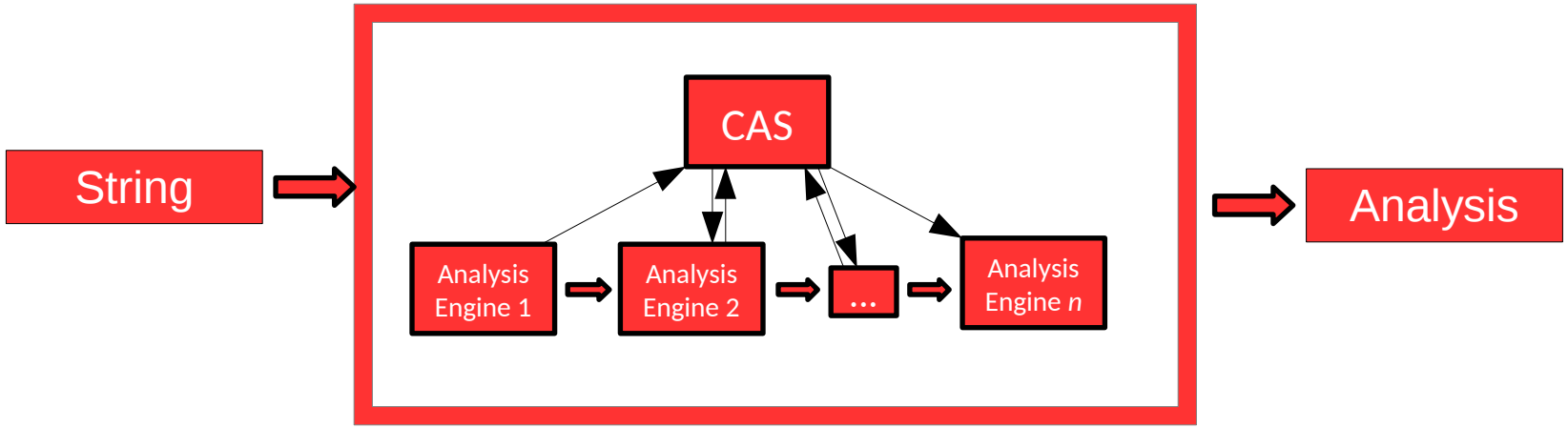


uimaFIT

- **F**actories
  - instantiate UIMA components without descriptor files
- **I**njection
  - remove boiler-plate code associated with configuration
  - Using Java annotations
- **T**esting
  - The initial use case for the library
  - Who wants to write descriptor files for unit tests?!



POJO Wrapper



myRdd.flatMap(text => analyzeText(text))

POJO Wrapper

- Constructor / initialize
  - AnalysisEngine myAE = createAnalysisEngine()
  - CAS myCAS = myAE.newCAS()
- analyzeText(myText)
  - myCAS.reset()
  - myCAS.setDocumentText(text)
  - myAE.process(myCAS)
  - extract values from CAS and return
-

...one more problem

- UIMA components are not serializable
  - Haven't tried Kryo yet...
- Workaround
  - *transient* analysis engine
  - readObject
    - Calls initialize method

Example Code

- OpenNLP provides UIMA wrappers
- Example code provides
  - POJO wrapper around OpenNLP UIMA wrappers
  - Example Spark call of POJO wrapper

Thank you!

Questions?