Task-based Execution Engine for JBOWL

Peter Bednár, Peter Butka

CIT FEI TU Košice

WIKT 2008

Outline

- Introduction - JBOWL library
- JBOWL Extended Conceptual Architecture
- Design and implementation of Execution Engine
- Conclusions

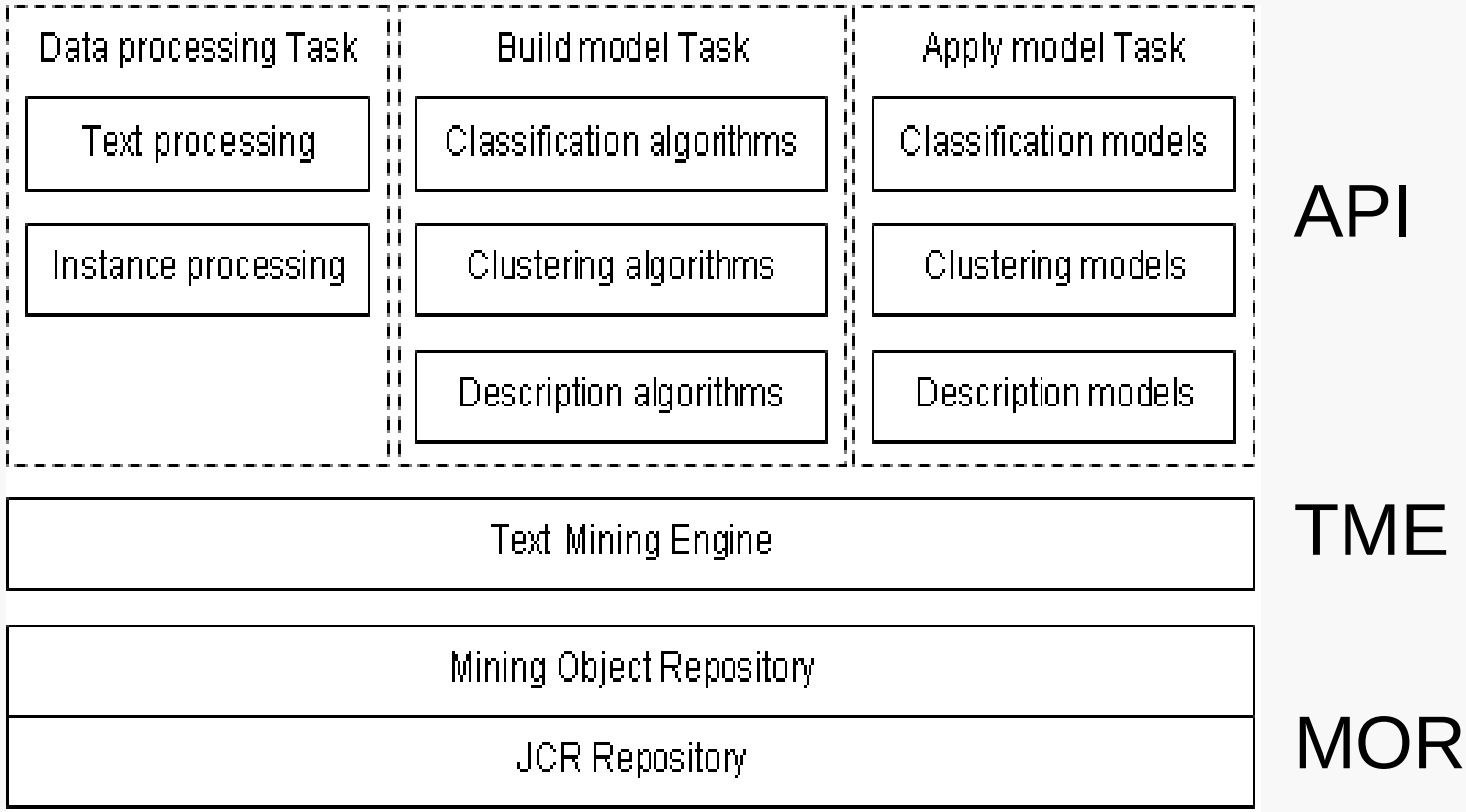
JBOWL Library - Introduction

- Java Bag-Of-Words Library (JBOWL)
- Functional requirements
  - efficiently preprocess potentially large collections of text documents with flexible set of available preprocessing techniques
  - adopted for various types and formats of text (e.g. plain text, HTML or XML).
  - text collections in different languages
  - support for indexing and retrieval in these text collections (and experiments with various extended retrieval techniques).
  - well-designed interface to knowledge structures such as ontologies, controlled vocabularies or WordNet
- Target groups
  - Text mining researcher (develop, test new text mining methods)
  - Application developers – use of API for building of WEB or GUI applications
  - Component developers – extensions or integrate of existing software with (part of) functionality of our framework
  - Students – students with basic understanding of the problems that text mining can solve.

JBOWL Library – Conceptual Architecture

- Java Data Mining API (JSR73) – architecture has three base components that may be implemented as one executable or in a distributed environment
  - **Application Programming Interface (API)**
    - user-visible classes and interfaces that allow access to services provided by the text mining engine (TME)
    - application developer using JBOWL requires knowledge only of API
  - **Text Mining Engine (TME)**
    - offers a set of text mining services to its API clients, manages execution of text mining tasks and importing/exporting existing mining objects from and to MOR.
    - can be implemented as a local library or as a server of client-server architecture
  - **Mining Object Repository (MOR)**
    - TME uses a mining object repository which serves to persisting of text mining objects

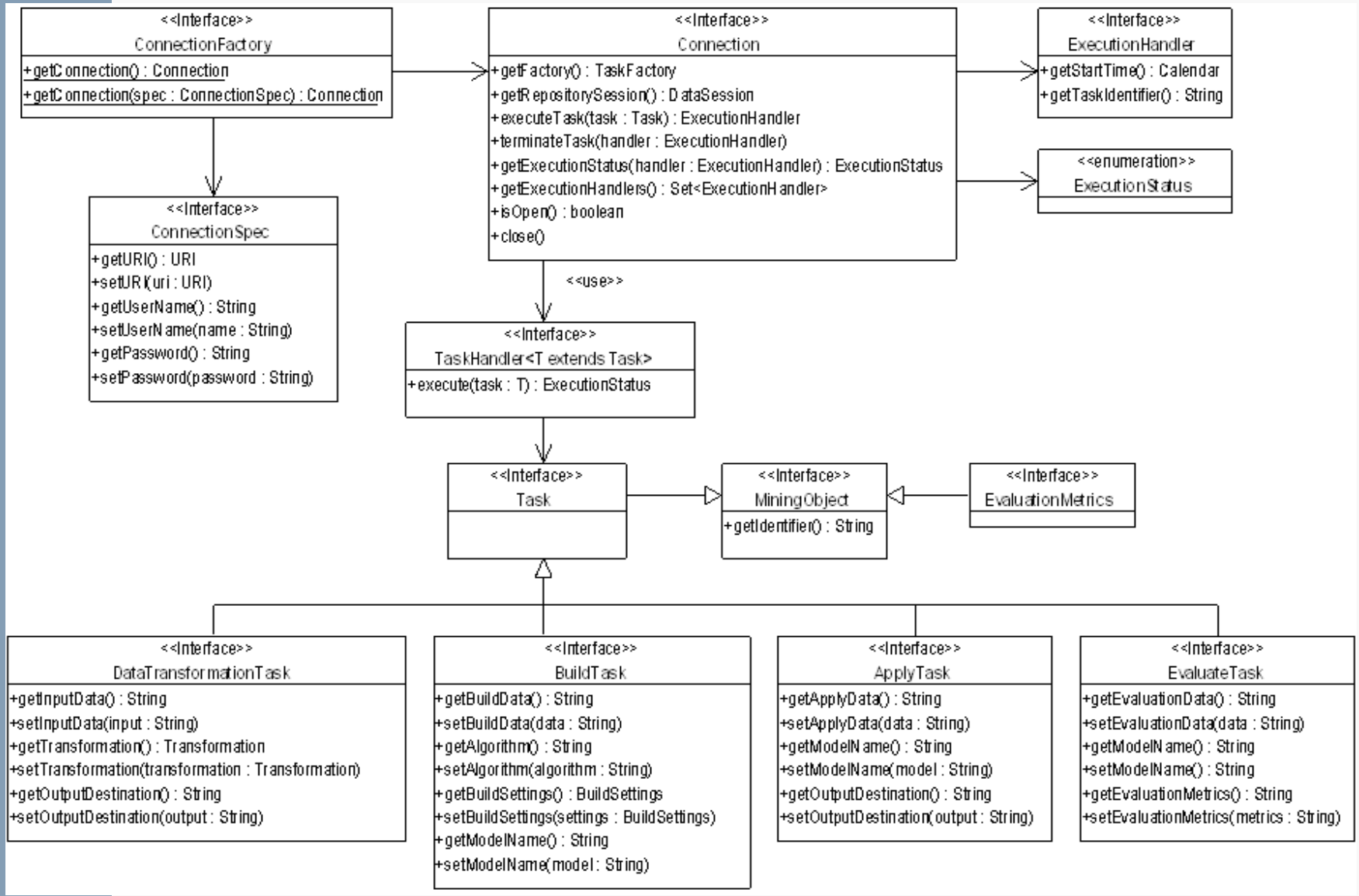
JBOWL Library – Extended Conceptual Architecture



Proposed Changes in Extended JBOWL

- Motivation
  - As-Is: local library execution, data and results locally, tasks and settings not fully distinguished for all text-mining tasks
  - To-Be: task-based engine for multithread/distributed execution of tasks, data and results available as content repository nodes, settings and tasks of different types transparent
- Extensions
  - Mining Object Repository
    - Use of Java Content Repository (JCR)
    - Integration of JBOWL and JCR – Mining Object Manager component maps Java objects to JCR nodes for serialization/de-serialization (similar idea like Hibernate)
  - TME – Execution Engine
    - Own (middleware-like) layer for running of tasks in multi-thread and potentially distributive manner, where
    - Developers are able to run tasks easily, do not need to know, where the tasks are really executed, they expect results and place where to found them
  - Tasks
    - Necessary changes regarding use of JCR for MOR implementation and new TME implementation

Execution Engine – Main Interfaces API



Execution Engine – Connection Interface

- **Connection**
  - Client obtains *Connection* object to TME (represent one text-mining session) in different way
    - directly without user authentication
    - registered on the client environment using the JNDI
  - Client specify details for connection specification
    - URI of the executed engine (if there are more TME instances)
    - username
    - password
  - *Connection* interface will allow client user to
    - obtain factory class to create new mining objects (i.e. data, tasks, build and task settings etc.)
    - obtain MOR session to save/load mining objects in MOR
    - execute, inspect and terminate text-mining tasks

Execution Engine – Task and TaskHandler

- **Task**
  - part of the client API, follow JavaBeans patterns for simple encoding of the objects in the remote protocols
  - specify all parameters required for the specific task, e.g. like references to the input data, path to output data, models, settings, …
- **TaskHandler**
  - Each type of the Task object has associated TaskHandler object responsible to execute this task
  - TaskHandler object creates execution process and perform all operations
    - e.g. Build Model Tasks task handler – load training data, create new instance of the algorithm specified as the task parameter, pass training data and build settings to the algorithm, produce new text mining model, store in the MOR on the path specified as the task parameter

Execution Engine – Connection Interface

- **Execution Handler**
  - After run of new thread for task => task is executed in the background
  - Client obtains Execution Handler object, which
    - identify running task
    - can be used to inspect Execution Status of the task process or to terminate task
- **Execution Status**
  - current status of the task execution

Conclusions

- JBOWL extension as internal and logical extension of library to
  - Support multi-thread running of tasks
  - Use of content repository paradigm for mining objects manipulation
  - Encapsulate different type of tasks to run in same fashion in TME
- Next steps
  - Finish implementation of core extensions
  - Update existing methods (if needed)
  - Testing and evaluation