Distributed Task-based Execution Engine for Support of Text-mining Processes in project PoZnaŤ

P. Butka, P. Bednár, F. Babič, K. Furdík
Centre for Information Technologies, Faculty of Electrical Engineering and Informatics, Technical University of Kosice, Boženy Němcovej 3, Kosice, Slovakia
peter.butka@tuke.sk, peter.bednar@tuke.sk, frantisek.babic@tuke.sk, karol.furdik@tuke.sk

Abstract — This paper describes design and implementation aspects for extension of our original software system developed in Java for support of information retrieval and text mining with specialized execution engine for different type of tasks. Some of our experiences and specific requirements of the real applications lead us to idea give the system possibility to run the tasks in distributive way. The result of the idea is task-based execution engine, which represents middleware-like transparent layer (mostly for programmers who want to re-use functionality of our package) for running of different tasks in multi-thread environment. The original system is being developed as open source with the intention to provide an easy extensible, modular framework for pre-processing, indexing and further exploration of large text collections. This specialized execution engine has been implemented within APVV project called PoZnaŤ (Support for knowledge creation processes) in order to extend proposed text-mining platform for educational and experimental purposes. Conceptual architecture of the system is provided as well as details regarding our extension of library like usage of content repository paradigm, representation and encapsulation of tasks, implementation of the engine itself and its role within PoZnaŤ platform.

I. INTRODUCTION

Our research and education goals in the area of text mining and information retrieval with the emphasis of advanced knowledge technologies for the semantic web resulted in design and implementation of library which is able to efficiently pre-process potentially large collections of text documents with flexible set of available pre-processing techniques, support various types and formats of text (e.g. plain text, HTML or XML), work with text collections in different languages (very different sorts of languages require significantly different approaches in pre-processing phase), 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.

The decision to design and implement a new tool, Java library for support of text mining and retrieval (with acronym JBOWL – Java Bag Of Words Library), was based on the detailed analysis of existing free software tools. More details regarding analysis, design and implementation of original library, as well as several text-mining methods already provided in the system (in time of preparing that paper) can be found in [1]. We will extend the description with some necessary details in next chapter.

Some of our experiences and specific requirements of the real applications lead us to idea give the system possibility to run the tasks in distributive way. This will give the programmers (application developers) also possibility to re-use library in more complex applications and domains. One example is our need to prepare education portal for text-mining for students on lectures of knowledge discovery from texts in domain of knowledge management. The original library has significant advantages in programming and incorporating of several text-mining methods for most of the problems in this complex domain like classification and clustering of documents, different pre-processing techniques or information extraction methods – this is more individually oriented usage of library, for simple one-problem running of tasks and experiments. On the other hand in our project (project PoZnaŤ) the goal is to provide portal for the students (lectures), where many users could run several text-mining tasks with different collections and evaluate the results. This type of application then logically needs to run tasks on machine in multi-thread or distributive way.

The result of this idea for extension is task-based execution engine, which represents middleware-like transparent layer (mostly programmers who want to re-use functionality of our package) for running of different tasks in multi-thread environment. In the next chapter we will introduce conceptual architecture of the system, some comments and ideas regarding our extension of library like usage of content repository paradigm, representation and encapsulation of tasks. Then design and implementation of the engine itself will be provided in more details. In the next chapter we will provide more information about broader architecture of the system which will use our engine in project as well as some details about project. At the end of the paper we will provide some future work remarks and conclusions.

II. CONCEPTUAL ARCHITECTURE OF JBOWL

JBOWL has the same architecture like standard Java Data Mining API (JSR 73 specification [3]). Also new specification is in preparing phase, but it is not finished yet, so we would stick to JSR73 (with extensions if some

1 http://web.tuke.sk/fei-cit/poznat/index-a.html
new concepts seem to be interesting for our purposes). This architecture has three base components that may be implemented as one executable or in a distributed environment:

- **Application Programming Interface (API)** - The API is set of user-visible classes and interfaces that allow access to services provided by the text mining engine (TME). An application developer using JBOWL requires knowledge only of the API library, not of the other supporting components.

- **Text Mining Engine (TME)** - A TME provides the infrastructure that offers a set of text mining services to its API clients. TME can be implemented as a local library or as a server of client-server architecture.

- **Mining Object Repository (MOR)** - The TME uses a mining object repository which serves to persisting of text mining objects.

TME manages execution of common text mining tasks, e.g. document analyzing, building a model, testing a model, applying a model on new data, computing statistics, and importing and exporting existing mining objects from and to MOR.

- **Data Processing Tasks** – input datasets (sets of textual documents) are processed using these tasks, it means that input documents are recognized, text in documents is tokenized and statistics are computed and saved, vector representation of documents (instances) is prepared according to statistics, scheme of weighting (tf-idf) is used for weighting of terms, filtering/pruning of terms is available like stop-words filtering, selection of terms, etc.

- **Build Model Tasks** – tasks here are responsible for building of appropriate model using chosen algorithm. Nice feature is that models and algorithms have very common structure and it is possible to re-use them for implementation of new algorithms and their combinations. Several text-mining or pre-processing approaches have been already implemented in our package:
  - Text categorization – several well-know algorithms like SVM, linear classifiers (e.g. perceptron), decision trees and rules induction, kNN, together with ensemble learning methods like boosting and bagging.
  - Clustering methods – kMeans, SOM and related methods (GHSOM), agglomerative clustering
  - NLP methods (especially for pre-processing steps) – ATN networks for deeper text analysis
  - Formal Concept Analysis (FCA) – usage of method itself, additionally hybridized with clustering and description methods
  - Description algorithms – Labelling of SOM models or FCA-based models using LabelSOM, extraction of keywords based on other approaches like analyzing of information gain, etc.
  - Meta-learning approach for text categorization

- **Apply Model Tasks** – models created by algorithms are applied to new data inputs in order to classify new documents, find appropriate cluster to some document, find out contextually similar documents within near formal concepts, use meta-model for automatic choosing of algorithm in different domain, etc. Conceptually, also evaluation of models and their processing (with some evaluation metrics) can be seen in this set of tasks (although in implementation they will be differently modelled).

In order to achieve running of all tasks in multi-thread text-mining engine, it is necessary to model them appropriately, we will show our approach in part related to
concrete design and implementation of execution engine in chapter III.

B. Text Mining Engine

As it was already written, TME manages execution of common text mining tasks. It can be done as usage of local library or as running tasks in server-client architecture. Our main point is to extend JBOWL with its own transparent layer for running of tasks in multi-thread and potentially distributive manner, where application developers are able to run tasks easily and they probably do not need to know, where the tasks are really executed, they expect results and place where they can find them.

This is idea almost identical to grid computing paradigm. We have already some experiences with combination of JBOWL library and grid environment (e.g. distributed classification (decision trees) of documents on the Grid [5], parallel/distributed implementation of GHSOM [6] or FCA method [7], etc.), which also implies our decision to logically extend library with multi-thread distributive support. Our experiences with previously mentioned work have shown some requirements for the application developers like running tasks naturally in multi-thread way, re-use of existing results and datasets in more complex way (every mining object will be connected to node in content repository – see next chapter of the paper for details) and possible extension to prepare engine running on different machine without developer needs to change his code (only setup connection), in order to fulfill middleware-like architecture of application based on text-mining tasks.

C. Mining Object Repository

Mining Object Repository (MOR) is used as a persistent storage for the processed text content and all artefacts generated during the text-mining process. Persistent objects include annotations of analyzed texts, data and evaluation statistics, indexed instances, text mining models and task settings. The implementation of the MOR is based on the Java Content Repository (JCR) specifications, which provide seamless integration with the existing content repositories.

The main concept of JCR specification is Node, which has associated data Properties. Text content is stored in the properties of the string type, but JCR also supports other data types like dates, Boolean values, real and integer numbers or arbitrary binary content. Nodes are explicitly organized in the hierarchy, but it is possible to have reference properties to other nodes, which can be used to represent non-hierarchical relations. JCR specification provides strong support for search of the content. Queries can be specified with XPath expressions or in the dialect of the SQL language. Other features of JCR specification, which provide benefits for implementation of the MOR, include support of transactions and type system, which specify constrains for possible properties and sub-nodes.

In order to simplify integration of Jbowl and JCR, MOR contains Mining Object Manager component, which maps Java objects to JCR nodes for serialization/de-serialization. The mapping mechanism is generic and can be used to map arbitrary Java objects to JCR in the similar way, how the object-relational mapping frameworks (like Hibernate) are used to map Java objects to relational database.

III. DESIGN AND IMPLEMENTATION OF EXECUTION ENGINE

API (Application Programming Interface) of the Execution Engine is divided to client part and server part to simplify implementation of remote task execution. The main interfaces of the Execution Engine are depicted on the following diagram (Figure 2).

A. Connection

To start working with Text Mining Engine (TME), client has to obtain Connection object which represent one text-mining session. Connection can be obtained in various ways, for example it can be created directly without user authentication or registered on the client environment using the JNDI. Client can specify details for connection specification like URI of the executed engine in the case that there are more TME instances, user name and 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 or load mining objects stored in the Mining Object Repository.
- Execute, inspect and terminate text-mining tasks.

B. Task and TaskHandler

API for tasks is divided to interfaces for task specification (Task interface) and for task execution (TaskHandler interface). Task objects are part of the client API and follow JavaBeans patterns, which allow simple encoding of the objects in the remote protocols. Task objects specify all parameters required for the specific task, like references to the input data, path where to store output data or models and all associated settings (build settings for algorithms, settings for data processing and settings for evaluation metrics).

Each type of the Task object has associated TaskHandler object, which is responsible to execute this task. According to parameters specified in the Task object, TaskHandler object will create new parallel execution process and perform all operations defined for the task. For example for the Build Model Tasks task handler will
load training data, create new instance of the algorithm specified as the task parameter, and pass training data and build settings to the algorithm to produce new text mining model. Model is then stored in the MOR on the path specified as the task parameter.

C. Execution Handler and Execution Status

When the Execution Engine creates thread for new task and execute task handler, the client invocation of the Execution Engine is immediately finished and task is executed on the background. Execution Engine will return to the client Execution handler, which identify running task and can be used to inspect execution status of the task process or to terminate task.

IV. PoZnaŤ

Project PoZnaŤ [2] aims at creation of new integrated platform for building a repository of explicitly represented shared objects in different formats for education and experimental purposes. The second part of developed platform represents a suite of software NLP (Natural Language Processing) tools, interconnected to ontology-based knowledge representations and semantic structures.

PoZnaŤ is based on running FP6 IST project called KP-Lab" (Knowledge Practices Laboratory) that is an ambitious project focused on developing a virtual learning system aimed at facilitating innovative practices of sharing, creating and working with knowledge in education and workplaces.

A. KP-Lab

The objective of the KP-Lab project is to develop theories, tools, and practical models to elicit deliberate advancement and the creation of knowledge, as well as the corresponding transformation of knowledge practices in education and workplaces.

Theoretical foundation of this project is trialogical learning approach. This approach has emerged from two others that have been adopted before in broader sense, i.e. monological and dialogical learning [9].

There are some other pedagogical approaches which are quite near the trialogical approach to learning, e.g. knowledge building, project-based learning, learning by design or progressive inquiry learning. Trialogical approach can be used to give a new focus on these existing pedagogical approaches. It is not tied to any of them but aims at given methods and features for supporting aspects of collaborative knowledge creation.

Technological goal of this project is represented by the KP-Lab System (modular, flexible and extensible solution) that consists of KP-Lab platform and the end-user virtual learning environment (KP-environment) with several integrated tools for collaborative knowledge practices (working with shared knowledge objects, managing knowledge processes, multimedia annotation, meeting support, visual modeling, etc.).

Relevant outcomes of the KP-Lab project have very limited possibilities of applicability in the Slovak conditions, since there is no specific support for Slovak language. TUK team from CITŤ is mainly involved in technological workpackages and tasks so adaptation of interesting features to our local conditions is the main background idea behind PoZnaŤ project.

The reason is that core of used knowledge management techniques, as Text Mining algorithms, indexing and annotation of knowledge artefacts, creation and maintenance of ontologies, are strongly language-dependent procedures.

B. PoZnaŤ objectives

- Development of integrated suite of tools for NLP in Slovak language

It's necessary to adapt existing tools developed within KP-Lab project to provide possibilities for processing of textual and multimedia documents written in Slovak language, to extract knowledge fragments from them, to integrate them and form them into resulting knowledge shared objects and processes. Design and development is oriented with correspondence to particular language levels, e.g. morphology, derivatology (word-formation), syntax (both deep and surface), and semantics, with relations to the existing structure of knowledge representation. Particular tools will be accessible as web services through specialized web portal[4].

- Development of data repository within suitable CSM[5] solution

The data repository is a core of the proposed system and consists from a corpus of training texts, and from data structures needed for particular phases of processing on the language levels. Several resources are already available, however, there will be adapted and integrated them by unifying the format (based on XML, compatible with standards recommended in the field for corpus linguistics [8]).

- Analysis of system outputs within methods of Artificial Intelligence

Outputs of the proposed system will be in form of particular courses, structures of goal-oriented knowledge shared objects, actions and activities, explicit representations of shared knowledge in different stage of elaboration. These outputs will be very useful further to analyse within several suitable approaches, e.g. Artificial Intelligence methods, e.g. using adaptable classification and heuristic clustering. The result of these analyses will be used as qualitative and quantitative evaluation within pilot applications.

- Evaluation of project outcomes within pilot application

Project outcomes, i.e. particular software components as well the integrated system will be verified and evaluated on the pilot application.

---

2 http://www.kp-lab.org/
3 http://web.tuke.sk/fei-cit/index-a.html
4 http://cit.fei.tuke.sk:8080/textminingweb/login.jsp
5 http://www.cmsmatrix.org/
within the course of Knowledge Management held on Department of Cybernetics and Artificial Intelligence, at Technical University of Košice. During this course students realize predefined tasks in implemented portal so this situation requires possibility to run tasks on machine in multi-thread or distributive way.

C. Architecture of the whole platform

The architecture of whole platform that has been implemented within PoZnaŤ project is depicted on the Figure 3.

![Figure 3. Architecture of whole platform with integrated parts.](image)

- The new implemented parts are Tools for NLP of Slovak language and Execution engine.
- The other platform parts have been developed within KP-Lab project and now they have to be redesign and adapt to new conditions and requirements.
- Management console is implemented as web portal, see Figure 4.

![Figure 4. Welcome screen of management console.](image)

The core of implemented platform is the ontology-based semantic representation of e-Learning entities and processes. Shared objects/entities and processes are semantically described, i.e. annotated, by the metadata – ontology concepts. It means that the concepts from the e-Learning ontologies are the key elements of system structure. Process of definition, creation, maintenance, and usage of semantic metadata depends on the content (meaning) of shared objects and processes, which can be extracted from the textual (multimedial) description.

The textual description of the objects and processes is language-dependent, so it is necessary to use NLP tools for processing entities described in Slovak language. To create adequate semantic metadata descriptions, the NLP tools should provide a good quality of meaning extraction from texts, as well as proper level of automation, with minimum of required administrative interventions.

In this process, and also during the qualitative evaluation of processes, the mechanisms of Text Mining (classification, clustering, and others) will be applied. Good results can be achieved by combining Text Mining with NLP methods.

D. Platform components

- **Tools for NLP of Slovak language** – data repository (dictionaries, text corpora, data resources for language analysis), algorithms of analysis (for each of language levels).
- **E-Learning ontologies** – semantic base for knowledge processes, metadata definition.
- **Data Mediator** – communication interface of web services.
- **Text Mining** – classification and clustering algorithms, used for information extraction from texts.
- **Annotation Services** – mechanisms for semantic description of knowledge shared objects by proper metadata.
- **Analysis of outputs, evaluation** – qualitative evaluation of system functionality, presentation of created knowledge structures for users.
- **Management console** – web-based application for setting global parameters, for external maintenance and visualisation of NLP and Text Mining algorithms.
- **Execution Engine** - middleware layer for running of different tasks in multi-thread environment.

V. CONCLUSION

In this paper we have introduced task-based execution engine middleware extension of JBOWL for support of multi-thread/distributed running of text-mining tasks. Important features of the extension is usage of Java Content Repository in Mining Object Repository as basic space for persisting of mining objects like documents and created models. This allows text mining engine layer (Execution Engine) to work with objects in more flexible way and then run (conceptually organized and encapsulated) tasks more easily in parallel/distributed multi-thread way.

More practically, implementation of our Execution Engine as internal and logical component of JBOWL provides support for wide types of applications. In our case engine will be tested in our project, where JBOWL is used as a main text-mining engine behind education portal for students in their study and experimenting with process
of knowledge discovery in texts (lectures related to knowledge management technologies).

ACKNOWLEDGMENT

The work presented in the paper is supported by the Slovak Research and Development Agency under the contract No. RPEU-0011-06 (project PoZnať) and by the Slovak Grant Agency of Ministry of Education and Academy of Science of the Slovak Republic within the project No. 1/4074/07 "Methods for annotation, search, creation, and accessing knowledge employing metadata for semantic description of knowledge" and project No. 1/3135/06 "Methods and tools for design of the integrated distributed applications based on ambients – higher-level agents".

REFERENCES


