D5.3.2 Real-time Stream Media Processing Platform and Cloud-based Deployment - v2

Alex Simov (Ontotext)

Abstract

FP7-ICT Strategic Targeted Research Project TrendMiner (No. 287863)
Deliverable D5.3.2 (WP 5)

This document presents the final version of the TrendMiner integrated platform. It focuses on the new developments and improvement of the platform since M24. This includes: integration of new components; updates of existing components; customizations for specific use-cases scenarios (finance) and adapting the system to be used in more general context (extended use-cases).

Keyword list: Integration, REST, scalability, cloud computing

Nature: Prototype
Contractual date of delivery: M36
Reviewed By: Márton Miháltz (RILMTA); Thierry Declerck (DFKI)
Web links: n/a
TrendMiner Consortium

This document is part of the TrendMiner research project (No. 287863), partially funded by the FP7-ICT Programme.

DFKI GmbH
Language Technology Lab
Stuhlsatzenhausweg 3
D-66123 Saarbrücken
Germany
Contact person: Thierry Declerck
E-mail: declerck@dfki.de

University of Southampton
Southampton SO17 1BJ
UK
Contact person: Mahensan Niranjan
E-mail: mn@ecs.soton.ac.uk

Internet Memory Research
45 ter rue de la Révolution
F-93100 Montreuil
France
Contact person: France Lafarges
E-mail: contact@internetmemory.org

Eurokleis S.R.L.
Via Giorgio Baglivi, 3
Roma RM
00161 Italia
Contact person: Francesco Bellini
E-mail: info@eurokleis.com

University of Sheffield
Department of Computer Science
Regent Court, 211 Portobello St.
Sheffield S1 4DP, UK
Tel: +44 114 222 1930
Fax: +44 114 222 1810
Contact person: Kalina Bontcheva
E-mail: K.Bontcheva@dcs.shef.ac.uk

Ontotext AD
Polygraphia Office Center fl.4,
47A Tsarigradsko Shosse,
Sofia 1504, Bulgaria
Contact person: Atanas Kiryakov
E-mail: naso@sirma.bg

Sora Ogris and Hofinger GmbH
Bennogasse 8/2/16
1080 Wien Austria
Contact person: Christoph Hofinger
E-mail: ch@sora.at

Hardik Fintrade Pvt Ltd.
227, Shree Ram Cloth Market,
Opposite Manilal Mansion,
Revdhi Bazar, Ahmedabad 380002
India
Contact person: Suresh Aswani
E-mail: m.aswani@hardikgroup.com

DAEDALUS - DATA, DECISIONS AND LANGUAGE, S. A.
C/ López de Hoyos 15, 3º, 28006 Madrid,
Spain
Contact person: José Luis Martínez Fernández
Email: jmartinez@daedalus.es

Institute of Computer Science Polish Academy of Sciences
5 Jana Kazimierza Str., Warsaw, Poland
Contact person: Maciej Ogrodniczuk
E-mail: Maciej.Ogrodniczuk@ipipan.waw.pl

Universidad Carlos III de Madrid
Av. Universidad, 30, 28911, Madrid, Spain
Contact person: Paloma Martínez Fernández
E-mail: pmf@inf.uc3m.es

Research Institute for Linguistics of the Hungarian Academy of Sciences
Benczúr u. 33., H-1068 Budapest, Hungary
Contact person: Tamás Váradi
Email: varadi.tamas@nytud.mta.hu
Executive Summary

In this document we present the final version of the TrendMiner integrated platform. It focuses on the new developments and improvement of the platform since M24. This includes:

- integration of new components;
- updates of existing components;
- customizations for specific use-cases scenarios (finance);
- and adapting the system to be used in more general context (extended use-cases).

The whole platform has been deployed on the Amazon EC2 Cloud\(^1\) and it is accessible for all consortium members.

---

\(^1\) [http://aws.amazon.com/ec2/](http://aws.amazon.com/ec2/)
Contents

Executive Summary ............................................................................................................. 3
Contents ............................................................................................................................... 4
List of abbreviations .......................................................................................................... 4
List of figures ....................................................................................................................... 5
1 Introduction ....................................................................................................................... 6
2 Integrated Platform Overview ........................................................................................... 7
3 New components integration .............................................................................................. 7
  3.1 Mimir Integration ............................................................................................................ 7
    Mimir Overview ............................................................................................................... 7
    Mimir Integration ............................................................................................................ 8
  3.2 Clustering Service Integration ....................................................................................... 9
    Clustering overview ......................................................................................................... 9
    Clustering management layer & data workflow .............................................................. 9
    New Clustering Service API .......................................................................................... 10
    RDF model for clusters ................................................................................................. 11
    Clustering Browser (UI) ................................................................................................. 12
  3.3 Summarization Service Integration ............................................................................... 12
4 Extended Use-cases Architecture .................................................................................... 13
  4.1 Data model for social media resources ........................................................................ 14
  4.2 New APIs for direct data warehouse management .................................................... 16
    RDF data import ........................................................................................................... 16
    RDF repository reset .................................................................................................... 16
5 Use-cases demonstrators deployment ............................................................................... 17
6 Extension for financial data support (WP6) ...................................................................... 17
  6.1 CSV Import Service API .............................................................................................. 17
  6.2 Data access APIs ............................................................................................................ 18
7 Cloud Deployment ............................................................................................................. 19
Conclusions .......................................................................................................................... 19

List of abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UX</td>
<td>User Experience</td>
</tr>
<tr>
<td>REST</td>
<td>REpresentation State Transfer</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>MG4J</td>
<td>Managing Gigabytes for Java</td>
</tr>
</tbody>
</table>
List of figures
Figure 1. TrendMiner overview ................................................................. 6
Figure 2. TrendMiner integrated platform ...................................................... 7
Figure 3. Mimir life cycle ............................................................................. 8
Figure 4. Clustering management components ............................................. 10
Figure 5. Clustering RDF model ................................................................. 11
Figure 6. Clustering browser ...................................................................... 12
Figure 7. Common architecture for the extended use-cases ....................... 13
Figure 8. Financial data and social media activity ....................................... 17
1 Introduction

TrendMiner provides a platform for distributed and real-time processing over social media streams. The platform covers all the phases from the social media stream processing lifecycle: large scale data collection\(^2\), multilingual information extraction and entity linking, sentiment extraction, trend detection, summarization and visualisation (Figure 1).

![Figure 1: TrendMiner overview](http://www.trendminer-project.eu/images/d5.3.1.pdf)

This document represents the updates of the platform since M24 and familiarity with D5.3.1\(^3\) is strongly encouraging. For the sake of brevity the general requirements and tasks sections are omitted here. The document focuses on the final implementation of the platform.

---

\(^2\) Data Collection and storage in TrendMiner is the focus of T5.1, and described in D5.1.1 “Real-time Stream Media Collection, v.1”

\(^3\) [http://www.trendminer-project.eu/images/d5.3.1.pdf](http://www.trendminer-project.eu/images/d5.3.1.pdf)
2 Integrated Platform Overview

From a higher level point of view there are no significant changes in the architecture since M24 prototype (Figure 2). The general workflow stays also more or less the same; however significant effort has been made for internal optimisations of the platform performance.

The new features of the current version of the platform include:
- Integration of new components (the Mímir framework)
- Updates of existing components supporting extended features (clustering, summarization)
- Opening the platform for third party contributing components (extended use-case partners tools)

![Figure 2: TrendMiner integrated platform](image)

3 New components integration

3.1 Mímir Integration

Mímir[^4] is an integrated semantic search framework, which offers indexing and search over full text, document structure, document metadata, linguistic annotations, and any linked, external semantic knowledge bases. It supports hybrid queries that arbitrarily mix full-text, structural, linguistic and semantic constraints. Its key distinguishing feature is the containment operators that allow flexible creation and nesting of full-

[^4]: http://gate.ac.uk/mimir/
text, structural, and semantic constraints, as well as the support for interactive knowledge discovery.

To improve the suitability of Mímir for large amounts of streaming data, we implemented live indexes that can accept new documents for indexing at the same time as they serve queries based on the documents already indexed. Mímir relies on MG4J\(^5\) for the implementation of on-disk indexes, and MG4J indexes are designed to be read-only, in order to maximize performance. Given these constraints, the approach we used was to index incoming documents in batches, and make each batch searchable as soon as it’s produced.

![Mímir Integration](image)

**Mímir Integration**

The integration of Mímir with the rest of the platform can be achieved at two levels - data provisioning and data querying & visualization. For the latter we decided to have a lightweight integration approach and to keep the interaction with the two existing use cases frontends independent. Still the user is presented a single entry point to the system and the option to choose which sub-system to use. The Mímir UI offers sophisticated search mechanisms over various aspects of the data but this comes at the price of certain expertise required by the user. The TrendMiner UI on the other hand

---

\(^5\) See [http://java-source.net/open-source/search-engines/mg4j](http://java-source.net/open-source/search-engines/mg4j) for more details.
D5.3.2 Real-time Stream Media Processing Platform and Cloud-based Deployment v2

provides simplified means for data exploration based entirely on UI widgets and controls. On the representational level the strength of Mímir is to deliver efficiently results from arbitrary complex queries whereas TrendMiner UI focuses on the analytical aspect of the data (aggregation and summarization) providing drill-down data exploration support.

The data provisioning integration is approached by extending the metadata pre-processing components to be able to send the final results for indexing to Mímir. The components interaction is based on the pre-existing HTTP API Mímir exposes for document indexing. From the platform side, the results from all pre-processing components are packaged as GATE document and passed to the API endpoint URL. This way any new data entering the system is synchronised immediately with the Mímir indexes.

3.2 Clustering Service Integration

Clustering overview

Spectral clustering is a clustering algorithm that has been shown to achieve state-of-the-art performance for a range of tasks, from image segmentation to community detection. This method treats the clustering problem as one of graph partitioning on the similarity graph between objects. The algorithm projects the objects via Singular Value Decomposition into a reduced space which aims for maximal separation of clusters. This way, spectral clustering is useful when data dimensionality is high. It is also particularly useful when the clusters can’t be discovered using a spherical metric, as is the case, for example, with k-means. The algorithm is also appealing because it is supported by spectral graph theory. More details on the algorithm and its implementation are available in D3.2.16.

The code of the clustering implementation and its packaging as a RESTful webservice is available as open source at github7.

Clustering management layer & data workflow

In order to enable the end users to utilize the complete power of the clustering service implementation we built an additional layer of components supporting various clustering tasks accessible directly through the UX layer. Such tasks include:

- constructing new clustering sets based on custom parameters;
- exploring clustering results;
- running parallel clusterings;
- sharing or removing clustering results.

The following diagram (Figure 4) reveals the major functional components of the clustering management layer. All activities and tasks are initiated from the UI frontend, refined by the UX Data Service component. Tasks related to data exploration are driven mainly by a Clustering Manager responsible for fetching the proper data from the data warehouse and delivering it in suitable form for visualization.

---

6 http://www.trendminer-project.eu/images/d3.2.1.pdf
7 https://github.com/danielpreotiuc/trendminer-clustering
The process of creating new clustering sets involves more components playing their role in the complex data processing chain. Here the role of the Clustering Manager is to trigger the proper clustering task described as a set of parameters and to orchestrate the whole subsequent process. The clustering task instructs the Data extractor how to collect the input data for clustering based on parameters like type of sources & time frame. The extractor collects the desired data and transforms it into the format the spectral clustering implementation requires it. The result is handled to the RDFizer component which parses it and transforms it in a format suitable for efficient consumption by the rest of the components. The actual model and representation is discussed in the following sections. The final result is loaded in the data warehouse which makes it immediately available to the UX.

The clustering computation task can be arbitrarily complex (the user can select an arbitrary amount of input data) which might cause certain performance issues for the platform. To cope with this possible problem, we built a service around the components involved, thus decoupling them from the rest of the components. The communication and workflow is implemented in an asynchronous way, utilizing tasks queues to handle arbitrary spikes of load. If necessary the service can further be spawn on cluster of computing nodes/machines to avoid processing bottlenecks.

The service API specification is described in the following section (New Clustering Service API).

![Figure 4: Clustering management components](image)

**New Clustering Service API**

The following table provides API specification of the integrated clustering creation service, comprising the source data selection, actual clusters computing and transforming the results into the RDF model data (next section).

<table>
<thead>
<tr>
<th>URL</th>
<th>/trendminer/clustering</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP Method</td>
<td>POST</td>
</tr>
</tbody>
</table>
| Parameters   | **sesameURL** - the location of the source data and the result storage location  
**source**: filter by source (twitter, facebook, news, blogs)  
**fromDate**: Start of the selected period (YYYY-MM-DD)  
**toDate**: End of the period (YYYY-MM-DD) |

10
The clustering computation in general can be a heavy and time consuming task therefore the service invocation is done asynchronously. On service request, all input parameters are validated, a clustering computation task is scheduled for execution and the service generates a successful response. The actual computation is started as soon as any preceding tasks are completed.

**RDF model for clusters**

In order to represent the results from the clustering computation and to have flexible means for managing them, we define a simple light-weight conceptual model for representing clusters and their properties. The following diagram represents the main classes with their properties and relations.

![RDF model diagram](image)

Based on the results produced by the spectral clustering service implementation, we identified three major types of entities:

- *ClusterSet* - a labelled container for clusters resulted from a single clustering service invocation. Each cluster set can be opened from exploration in the UI.
- *Cluster* - a labelled set of terms, forming a common topic
- *Word* - representation of a single term within a cluster with its relevance score.

On the ground level this model is represented as RDF graphs and stored in the RDF metadata store. Any operation on the clustering model is based on querying and updating the RDF data. For demonstrative purposes we include a fragment of clustering data represented in RDF (Turtle):

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix tm-cluster: <http://trendminer.eu/cluster#>.
<http://trendminer.eu/usecases/sora/cs_sora2013> a tm-cluster:ClusterSet ;
 rdfs:label "Twitter data april 2013" ;
 tm-cluster:hasCluster <http://trendminer.eu/usecases/sora/cs_sora2013/cl_1> ;
 tm-cluster:hasCluster <http://trendminer.eu/usecases/sora/cs_sora2013/cl_2> ;
 tm-cluster:hasCluster <http://trendminer.eu/usecases/sora/cs_sora2013/cl_3> ;

<http://trendminer.eu/usecases/sora/cs_sora2013/cl_1> a tm-cluster:Cluster ;
 rdfs:label "1" ;
 tm-cluster:hasWord <http://trendminer.eu/usecases/sora/cs_sora2013/cl_1/%23arnautovic> ;
 tm-cluster:hasWord <http://trendminer.eu/usecases/sora/cs_sora2013/cl_1/%23autfar> ;
 tm-cluster:hasWord <http://trendminer.eu/usecases/sora/cs_sora2013/cl_1/%23autger> ;
```

---

8 This allows to consider a cluster as an instance that can be associated with any other object in the triple store.
3.3 Summarization Service Integration

The summarization service developed as part of WP4 (D4.1.2) has been integrated in the TrendMiner platform providing ranking facilities over the user data of interest (tracks). The application of the service is initiated by the user over the current result set to deliver the topmost representative tweets or other types of resources.
The service itself exposes a RESTful API which accepts a collection of textual resources in Twitter JSON format and responds with a ranked list of the same resources. The input preparation includes extracting the textual content of the resources as well as any entities and topics detected in the text during the pre-processing phase. The result (ranking list) is applied directly to the representation layer without any further actions.

4 Extended Use-cases Architecture

The inclusion of the new use-cases during the third year caused certain changes on the architecture of the platform but as a whole the backbone of the system is preserved. No major changes of the general workflow were required. A significant direction of changes is related to improving the openness of the system. We exposed more APIs for accessing and management of different sub-parts of the system as well as we built cleaner and more formal models for resources and other moving parts of the platform. The rationale for these changes is that the text mining tools developed by the second year of the project do not completely cover the languages required by the new partners. On the other hand the new partners came with their own means for handling these specific tasks as well as custom data collection approaches (different source and selection criteria).

Thus a reasonable integration approach was to replace the whole text pre-processing chain until the point where the language specifics are abstracted for the processing components (for example spectral clustering, text summarisation, ontology-based search).

Having analyzed case-by-case the needs and the abilities of each of the use-case partners, we identified the entry point of the data warehouse as the most appropriate border line between use-case specific tools and the common workflow components. The following Figure 7 represents the common architecture applicable to the extended use-cases workflow.

![Figure 7: Common architecture for the extended use-cases](image-url)
In order to implement this architecture along with the mainstream TrendMiner platform we had two major tasks:

- clean formal model for the resources entering the data warehouse
- exposing the data warehouse as independent component with appropriate access APIs

### 4.1 Data model for social media resources

Defined as an abstract model in the previous version of this deliverable here we provide more details on the actual data modelling and representation. Previously this model was internal for the platform, so the actual details were not essential. Now, as we provide direct access to the data warehouse, detailed descriptions here play an important role for anyone willing to publish annotated resources to the platform. For more coherent description we start with a short summary of the model we presented in D5.3.1:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>identifier of the resource</td>
<td>11050918246</td>
</tr>
<tr>
<td>author</td>
<td>Author name (and URI if it can be resolved)</td>
<td>David MacLean</td>
</tr>
<tr>
<td>url</td>
<td>Original source on the Web</td>
<td><a href="http://original.web.pg">http://original.web.pg</a></td>
</tr>
<tr>
<td>topics</td>
<td>Related keywords</td>
<td>Labour, Budget, etc.</td>
</tr>
<tr>
<td>time</td>
<td>time of issuing or collecting the resource</td>
<td>2010-03-25T22:05:40</td>
</tr>
<tr>
<td>text</td>
<td>The actual text content of the resource</td>
<td>&quot;Budget 2010: Labour is stealing from our children's future to buy votes ...&quot;</td>
</tr>
<tr>
<td>source</td>
<td>the source the resource is retrieved from</td>
<td>Twitter, facebook, blogs, ...</td>
</tr>
<tr>
<td>sentiment</td>
<td>sentiment/polarity value</td>
<td>float number, negative or positive</td>
</tr>
<tr>
<td>location</td>
<td>geo-location related to the resource (location of origin)</td>
<td>![<a href="http://dbpedia.org/resource/London%3E">http://dbpedia.org/resource/London&gt;</a></td>
</tr>
<tr>
<td>language</td>
<td>&quot;en&quot;, &quot;de&quot;, ....</td>
<td></td>
</tr>
</tbody>
</table>

The concrete realization of this model represented in RDF is represented in the following table. We targeted at having a simple and comprehensible representation reusing popular and well accepted vocabularies. Thus our internal RDF model is based on the Dublin Core Metadata Initiative with some minor additions covering the specific modelling aspects. For readability improvement in the text we adopted the following namespaces:

- `dc-terms: <http://purl.org/dc/terms/>`
- `dc: <http://purl.org/dc/elements/1.1/>`
- `tm: <http://trendminer.eu/>`
- `xsd: <http://www.w3.org/2001/XMLSchema#>`

---

9 http://dublincore.org/
RDF model for **Social Media Resources:**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Range</th>
<th>Required?</th>
<th>Multi-valued?</th>
</tr>
</thead>
<tbody>
<tr>
<td>dc:identifier</td>
<td>Internal identifier for the resource</td>
<td>literal</td>
<td>optional</td>
<td>no</td>
</tr>
<tr>
<td>dc:originLocation</td>
<td>Original location on the Web (if available)</td>
<td>URI</td>
<td>optional</td>
<td>no</td>
</tr>
<tr>
<td>dc-terms:creator</td>
<td>The author of the resource</td>
<td>literal</td>
<td>required</td>
<td>no</td>
</tr>
<tr>
<td>dc-terms:subject</td>
<td>Keywords/topics associated with the resource</td>
<td>literal</td>
<td>optional</td>
<td>no</td>
</tr>
<tr>
<td>dc:hashtag</td>
<td>Any hashtags found in the text (including the hash sign)</td>
<td>literal</td>
<td>optional</td>
<td>yes</td>
</tr>
<tr>
<td>dc:sentiment</td>
<td>Preferably in the range [-1.0 - 1.0]</td>
<td>xsd:float</td>
<td>optional</td>
<td>no</td>
</tr>
<tr>
<td>dc:description</td>
<td>The original resource text</td>
<td>literal</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>dc:date</td>
<td>The time stamp of the resource creation</td>
<td>xsd:datetime</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>dc:source</td>
<td>The source the resource: twitter.com, facebook.com, blogs, news</td>
<td>literal</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>dc-terms:references</td>
<td>Mentions found in the text</td>
<td>URI</td>
<td>optional</td>
<td>yes</td>
</tr>
<tr>
<td>dc-terms:ref_lab</td>
<td>The literal surface level representation of the mentions (above). They should come in pairs</td>
<td>literal</td>
<td>optional</td>
<td>yes</td>
</tr>
<tr>
<td>dc:language</td>
<td>Language id: 'en', 'it', 'de', 'es' ...</td>
<td>literal</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>dc:location</td>
<td>The geo location of resource origin (if available)</td>
<td>literal</td>
<td>optional</td>
<td>yes</td>
</tr>
<tr>
<td>dc:location_uri</td>
<td>The same as above but resolved to URIs in DBpedia (if possible)</td>
<td>URI</td>
<td>optional</td>
<td>yes</td>
</tr>
<tr>
<td>tm:hasTokens</td>
<td>Whitespace separated list of tokens which might be normalized, lemmatized or whatever is necessary to feed the clustering tool properly.</td>
<td>literal</td>
<td>required for clustering tool only</td>
<td>no</td>
</tr>
</tbody>
</table>

Here follows an example representation of a single resource in RDF Turtle:

```turtle
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix dc-terms: <http://purl.org/dc/terms/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .

<http://trendminer.eu/resources#tweet_327367546749718528> dc:identifier "327367546749718528" ;
dc-terms:creator "waldydzikowski" ;
dc:description "@KrzysztofLisek Dziekuje, w koncu sie zdecydowalem;) Pozdrawiam" ;
dc:language "pl" ;
dc:sentiment "1.0"^^xsd:float ;
```
4.2 New APIs for direct data warehouse management

An important step in making the platform open for third party contributions is by providing means for direct access of the operational data in the warehouse. At the same time integrity and consistency of the system data should be guaranteed. To approach these requirements we provided limited access APIs on top of the data warehouse.

**RDF data import**

This service enables publishing of RDF data directly in the data warehouse (OWLIM) which is immediately available in the UI demonstrator. The service can be polled on regular bases to ensure live updates of the data.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>/tm-data-service-v3/rdf</td>
</tr>
<tr>
<td>Method</td>
<td>POST</td>
</tr>
<tr>
<td>Input Parameters</td>
<td>Content-Type - any valid RDF serialization format. Valid values are: application/rdf+xml, text/plain, text/turtle, text/rdf+n3, text/x-nquads, application/rdf+json, application/trix, application/x-trig, application/x-binary-rdf</td>
</tr>
<tr>
<td>Response</td>
<td>Success confirmation or error report (HTTP codes)</td>
</tr>
</tbody>
</table>

**RDF repository reset**

This service can be used to remove all the documents from the data store resetting it to its initial state. This functionality is supposed to be used mainly during the development phase by the use-case partners responsible for delivering their annotated data directly in the store.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>/tm-data-service-v3/admin/resetstore</td>
</tr>
<tr>
<td>Method</td>
<td>DELETE</td>
</tr>
<tr>
<td>Input parameters</td>
<td>none</td>
</tr>
<tr>
<td>Response</td>
<td>Success confirmation or error report (HTTP codes)</td>
</tr>
</tbody>
</table>
5 Use-cases demonstrators deployment

Since June 2014 we are running five separate platform deployments for each of the use-cases. Each of them is reachable at URL:

http://<partner_name>-uc.ontotext.com/TrendMiner/tracks.html

where <partner_name> is one of: sora, ek, ipipan, riltna, daedalus.

During the development phase each partner has separate credentials to avoid accidental damaging of someone else data. Each demonstrator is configured separately depending on the corresponding partner's needs (recall Figure 2 and Figure 7).

6 Extension for financial data support (WP6)

For the purpose of the financial use-case it is of great importance to be able to align stock market information with social media activities (Figure 8). In this way it is easy to analyze the correlation between prices and hot topics and trends over time.

The nature of the financial data is numeric tables containing stock exchange market data on daily bases (companies, prices, volumes, indexes, etc.). The data is represented as tabular format (CSV) files for which we had to provide a separate API for upload and querying.

![Figure 8. Financial data and social media activity](image)

### 6.1 CSV Import Service API

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value &amp; Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>/tm-anno-service/stockdata</td>
</tr>
<tr>
<td>HTTP Method</td>
<td>POST</td>
</tr>
<tr>
<td>Input Parameters</td>
<td>Content-Type - ‘text/csv’</td>
</tr>
</tbody>
</table>
**Company** - the name of the company of index for which the data is related

CSV content as request body. The format is three columns: date, volume, and price.

Example:

17/04/2014,10154800,524.94
16/04/2014,7670200,519.01

**Response**

Success confirmation or error report (HTTP codes)

---

### 6.2 Data access APIs

Listing all companies for which the platform has financial data uploaded.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value &amp; Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>/tm-data-service-v3/companies</td>
</tr>
<tr>
<td>HTTP method</td>
<td>GET</td>
</tr>
<tr>
<td>Input parameters</td>
<td>-</td>
</tr>
<tr>
<td>Response</td>
<td>JSON array of company names or error message</td>
</tr>
</tbody>
</table>

Accessing detailed information for a company in a certain time interval.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value &amp; Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>/tm-data-service-v3/companies/{company-id}/stockdata/</td>
</tr>
<tr>
<td>HTTP method</td>
<td>GET</td>
</tr>
<tr>
<td>Input parameters</td>
<td>company-id - the company for which data is requested from - start of time period to - end of time period Time format is: yyyy-MM-dd'T'HH:mm:ss</td>
</tr>
<tr>
<td>Response</td>
<td>JSON object containing the information required by the timeline diagram to plot the volumes and prices for the specified period</td>
</tr>
</tbody>
</table>
7 Cloud Deployment

The following table summarizes the cloud deployment resources consumed and the purpose for each resource. The described deployment setup is capable to serve all the five use-cases without significant performance drops. All machines are based on the Linux OS, though it's not a strict requirement.

<table>
<thead>
<tr>
<th>Machine type</th>
<th>Purpose</th>
<th>Price category</th>
<th>Number of instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>r3.2xlarge</td>
<td>Data warehouse, requiring plenty of RAM and fast I/O. Capable to serve the data for all use-cases on one host.</td>
<td>$$$</td>
<td>1</td>
</tr>
<tr>
<td>c3.2xlarge</td>
<td>Pre-processing tools (LODIE, Sentiment, Geo-location, etc.) requiring high CPU utilization and I/O, not so much RAM.</td>
<td>$$</td>
<td>2</td>
</tr>
<tr>
<td>c3.2xlarge</td>
<td>Clustering service - balance between CPU and RAM utilization requirements. Clustering computing is CPU intensive operation; however it is used relatively rarely.</td>
<td>$$</td>
<td>1</td>
</tr>
<tr>
<td>c3.2xlarge</td>
<td>UI &amp; Backend services - moderate CPU and Memory requirements, minimal I/O usage</td>
<td>$</td>
<td>5</td>
</tr>
<tr>
<td>r3.large</td>
<td>Mimir indexing and querying - relatively high CPU and RAM utilization</td>
<td>$$</td>
<td>1</td>
</tr>
</tbody>
</table>

Conclusions

In this document we have presented the final version of the TrendMiner integrated platform. It focuses on the new developments and improvement of the platform since M24. This includes integration of new components; updates of existing components; customizations for specific use-cases scenarios (finance); and adapting the system to be used in more general context (extended use-cases).

The whole platform has been deployed on the Amazon EC2 Cloud and it is accessible for all consortium members.