Social Semantic Emotion Analysis for Innovative Multilingual Big Data Analytics Markets

D3.4 Acceleration of Large-scale Emotion Analysis Methods, initial version

Project ref. no	**H2020 644632**
Project acronym	**MixedEmotions**
Start date of project (dur.)	**01 April 2015 (24 Months)**
Document due Date	**31 March 2016 (Month 12)**
Responsible for deliverable	**Paradigma Tecnológico**
Reply to	**jvmarcos@paradigmatecnologico.com**
Document status	**Final**

Project reference no.	**H2020 644632**
Project working name	MixedEmotions
Project full name	Social Semantic Emotion Analysis for Innovative Multilingual Big Data Analytics Markets
Document name	D3.4_Acceleration_InitialVersion (version 1)
Security (distribution level)	PU
Contractual delivery date	31 March 2016
Deliverable number	D3.4
Deliverable name	Acceleration of Large-scale Emotion Analysis Methods, initial version
Type	Other
Version	Final
WP / Task responsible	WP3 / Paradigma Tecnológico
Contributors	José Víctor Marcos Martín, Carlos Navarro De Martino, Gabriela Vulcu
EC Project Officer	Susan Fraser
Document Location	https://confluence.deri.ie:8443/display/mixedem/MixedEmotions+Deliverables+M1-M12

Index

Executive Summary

This document describes the first phase of task T3.2, which is devoted to the optimization of the modules composing the MixedEmotions platform. In this initial phase, an exhaustive analysis of the modules has been performed. It has revealed two different groups of platform components depending on their nature and specific attributes: non-distributed and distributed modules. Hence, distinct approaches have been designed for each of them in order to optimize their use (i.e., minimization of allocated resources and latency). For those modules in the former group, the integration in the platform has been proposed as an optimization approach. This integration involves the definition of a standard format to exchange information in the platform, which is given by the JSON-LD format. In case of distributed modules, parallelization is adopted as the main acceleration strategy.

Two implementations are considered in order to parallelize the operations of a processing module. The first one is given by the use of a map-reduce programming approach. In the platform, Spark is used as the framework for parallel data processing. Hence, the API provided by Spark can be used to implement a method according to this programming paradigm. On the other hand, software replication can be employed to obtain a parallelization scheme. This approach consists of the deployment of a given software module in different machines of the cluster. As a result, these instances can be used simultaneously.

The result from this task is the definition of a set of strategies for an efficient use of the tools provided by the MixedEmotions platform. In addition, the software to implement these strategies has been developed as a specific orchestrator. It is in charge of managing the distribution of the data between the machines in the cluster, as well as the identification of sets of subtasks from large processing tasks.

1. Introduction

This document is the second deliverable of WP3 “Big Data Platform Architecture and Data Analysis Acceleration”. It describes the strategies adopted for the integration and acceleration of the different modules composing the MixedEmotions platform. The results described in this deliverable are derived from the initial completion of the second task in WP3, which was entitled as “Dynamic Data Flows, Architecture Optimization and Platform Acceleration”.

Briefly, the purposes of this task are the following ones:

- To analyse the algorithms and data flows needed for the efficient processing, emotion analysis and data management of very large multimodal data sets.
- To propose suitable architectures, breakdown of tasks, and flow of data for acceleration of the time and memory critical algorithms.
- To design the architectures according to computational requirements and dynamic allocation of resources.
- To check the behaviour of the algorithms of interest on such architectures.

In the rest of the document, we provide an exhaustive description of the approaches adopted towards the completion of this task, with a focus on the objectives initially posed.

2. Analysis of the algorithms

2.1. Acceleration through parallelization

The MixedEmotions platform is intended for the processing of large volumes of multimodal data in order to enable the automatic and objective recognition of entities, sentiments and emotions from them. The outcomes derived from the platform could then be used to optimize the decision making process of a company based on a more accurate knowledge of their customers and partners.

In this scenario, one of the functionalities to be implemented in the MixedEmotions platform is the possibility of analyzing data from different sources, representing such a volume that cannot be easily handled by conventional applications. Therefore, the main approach to accelerate the execution of data processing in MixedEmotions will be **parallelization**.

The adopted strategy follows the divide-and-conquer pattern. It is employed in many sequential algorithms. According to this approach, a problem is solved by splitting it into a number of smaller subproblems, solving them independently, and merging the subsolutions into a solution for the whole problem. The subproblems can be solved directly, or they can in turn be solved using the same divide-and-conquer strategy, leading to an overall recursive program structure. Figure 1 illustrates the execution pattern when compared to the conventional sequential execution.

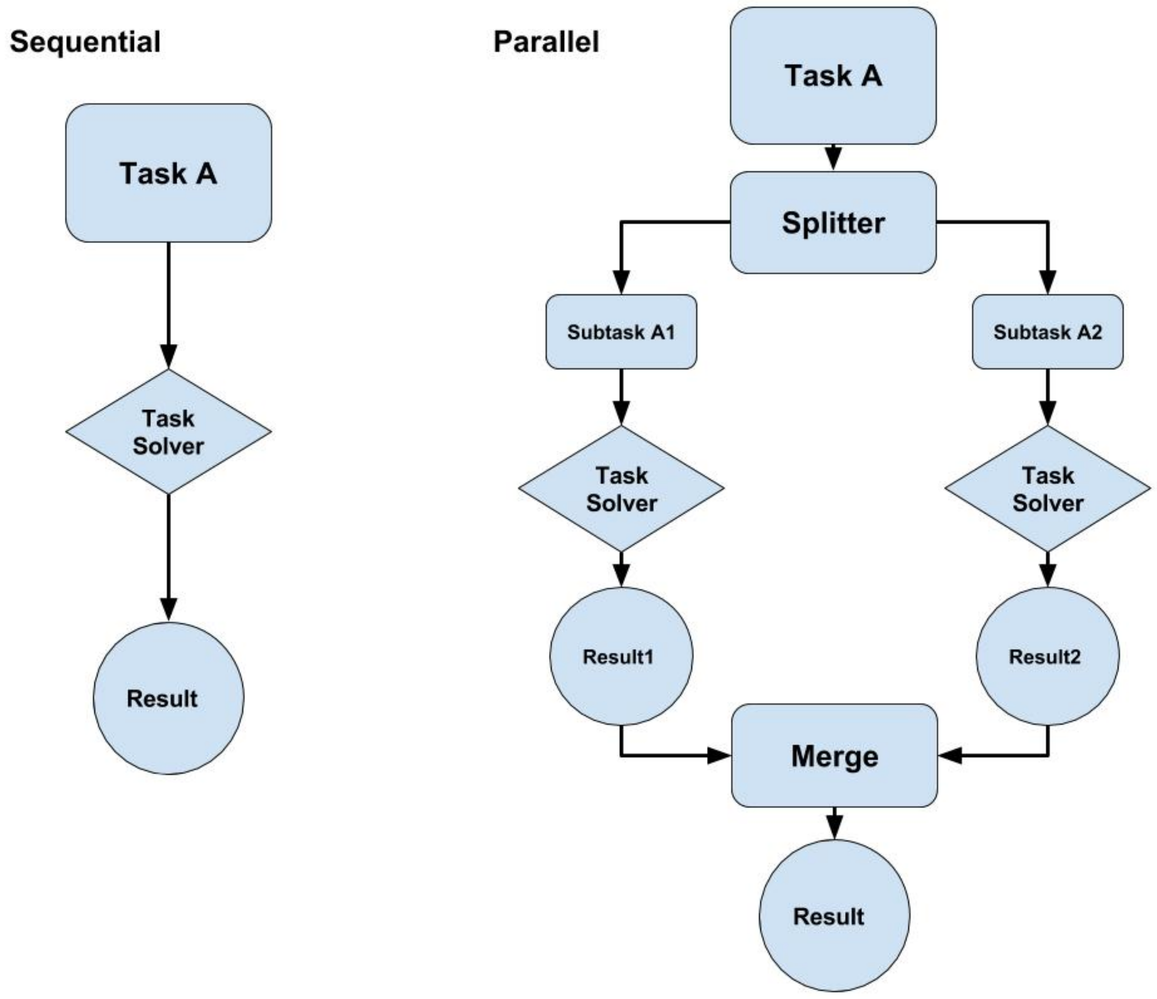


Figure 1. Scheme of the parallelization strategy adopted for the acceleration of the MixedEmotions modules.

As can be observed, the cost of the parallelization strategy is increased complexity. New agents must be included in our system. These correspond to the splitter, which is in charge of dividing the initial task into smaller ones, and the merge, which combines the results from different subtasks to get the global result.

On the other hand, parallelization will enable the **scalability** of the solution since distributed and parallelizable technologies suitably adapt to use cases with a growing demand for processing needs.

Parallelization of MixedEmotions functionalities involves two main technologies to be included in the architecture of the platform:

- Distributed data storage system. To process data in a parallel way, they must be conveniently stored in distributed and scalable storage systems. These technologies, which include tools such as HDFS, Cassandra, MongoDB or ElasticSearch, are usually accompanied by easy-to-use software that enables connecting capabilities. Hence, read/write operations from/to the database can be accomplished in an easy and efficient way, avoiding undesired bottlenecks.

- Distributed data processing. A specific framework for distributed data processing will be adopted. This software will be responsible for the definition of subtasks from a given large processing task (task splitting) and the coordination of the results for each of these subtasks (merging results). These actions correspond to the elements included in the system due to parallelization, as detailed in the previous scheme.

Once the inclusion of distributed technologies in the platform is assumed, strategies for the acceleration of the modules will be focused on how to get the most out of these technologies. To this end, the following points of the document describe the actions and strategies designed for the achievement of this goal.

2.2. Initial assessment of the MixedEmotions modules

As an initial step for this task, we analyzed each of the modules that compose the MixedEmotions platform. At this point, we identified two large groups, depending on their functionality:

1) Non-distributed processing modules. These modules enable concrete actions on the data, which are generally carried out at the beginning of the processing pipeline. In addition, due to their nature (i.e., the technology on which the module is based or the infrastructure required for its deployment), the parallelization of these modules is a complex task. As a result, non-distributed processing modules in the MixedEmotions platform are characterised by the following properties:

    a) Modules used in initial phases of the pipeline. Usually, these are involved in the preparation of the data for their subsequent analysis or are also in charge of collecting the data from external sources.

    b) Modules that are not applied to a large number of items. Non-distributed modules process data sequentially, as they do not have information on the size of the complete dataset or the number of individual entities to be processed.

    c) Modules that do not involve intelligent processing methods for the automatic interpretation of data. These modules are neither intended for the extraction of information from the data entity to be processed nor for the recognition of entities, sentiment or emotion. Thus, these modules do not require the application of trained models, dictionaries or taxonomies.

    In the MixedEmotions platform, non-distributed processing modules include those responsible for the following functionalities: speech-to-text conversion, translation, information crawlers (collecting data from Twitter, Facebook, websites or other databases), emotion recognition from audio/video and social network analysis.

2) Distributed processing modules. As a difference to those modules in the previous group, distributed processing modules provide advanced data processing capabilities in order to identify sentiments, recognize emotions and analyze network connections. These modules are equally applied to a large number of entities, so the computation of each entity does not depend on the results from other entities. therefore, processing can be directly carried out in a parallel manner. These modules are characterized by the following features:

    a) They perform advanced data processing for NLP or network analysis. Modules responsible for processing text in order to identify entities, concepts, sentiments or emotions will be initially run in parallel.

    b) The input to distributed processing modules will consist of a large amount of items with a small size (i.e., they suitably adapt to parallel processing). In addition, the exact size of the dataset to be processed can be known in advance, which makes the parallelization approach feasible.

The modules of the MixedEmotions platform that are offered in a distributed version are those in charge of the following actions: text analysis for entity, concept, sentiment and emotion recognition.

2.3. Approaches for MixedEmotions modules acceleration

The proposed division of the modules directly influences the definition of the platform architecture, as explained in deliverable D3.1. Briefly, non-distributed modules will be available as standalone applications. Hence, they will be hosted by a single machine, running in a local mode. On the other hand, distributed processing modules will be available as parallelizable tools. Then, it is the user who decides to use them for scalable and distributed processing or, on the contrary, to run them in a local mode. The adopted orchestrator will determine which of both modes will be used.

In this context, the following subtasks are identified in order to accomplish integration and acceleration of the modules in the platform:

- In the case of non-distributed modules, since they will run in a local mode, the integration will involve the definition of the format of both their inputs and outputs. In addition, the communication of these modules with other resources of the platform needs to be addressed. For instance, the capability for writing the output data from the module in the storage system of the platform is one of the integration and acceleration actions to be addressed. For the integration of these non-distributed modules, each of them has been thoroughly analysed. For instance, after the analysis of the crawlers, they were conveniently modified to provide the output in accordance with the format expected by the rest of the modules in the platform. In this case, a data model was defined to store the items to be processed. These are persisted in files, with each line corresponding to an ingested item.

- For distributed processing algorithms, we identified two different options for their integration and acceleration. First, the implementation of the modules on a distributed processing framework like Spark. This would involve a new implementation of the module, which may require a great effort in some cases. Therefore, the second option is to enforce a parallel approach by installing the module (i.e., the program or the library that provides the functionality) in every machine of the cluster. Hence, the Spark framework can be used to distribute the data and the processes between the machines. As a result, the processing capability of the initial standalone software is parallelized. The main challenge in this task is the integration of different technologies. It is worth noting that Spark is the tool used to distribute the data and orchestrate processing. However, standalone software may be written in different programming languages. To overcome this limitation, we will make use of the "pipe" tool provided by the Spark API. In the following list, we specify the acceleration strategy adopted for each of the analyzed modules in this initial phase of the task.

Functionality	**Language**	**Strategy**
Sentiment Extraction	English	Replication
Sentiment Extraction	English, Czech	Map-reduce programming
Sentiment Extraction	English, Spanish	Replication
Emotion Recognition	English, Czech	Replication
Emotion Recognition	English, Spanish	Replication
Entity Extraction	Spanish	Map-reduce programming
Entity Extraction	English	Replication
Topic Extraction	English	Replication
Topic Extraction	Spanish	Map-reduce programming
Entity Linking	English	Replication

Suggestion mining	English	Replication

Table 1. Strategies adopted for the optimization of some of the modules in the platform.

3. Influence of acceleration strategies on the design of the platform architecture

The initial version of the platform architecture described in deliverable D3.1 “Architecture Specification and Platform Implementation, initial version” was determined by the characteristic of the algorithms. In this preliminary version, the approaches suggested for the integration of the different modules, as described in the previous section, were taken into account. Hence, the distributed (or non-distributed) nature of the MixedEmotion platform modules was clearly specified.

The definition of the platform architecture aims at optimally accommodating each of the modules provided by the project partners. In addition, it aims at improving the performance of the modules when large amounts of multimodal data need to be processed. The main decisions on the design of the platform architecture directly influencing the acceleration of the modules and processing algorithms are the following ones:

- Spark-based orchestrator. As described before, distributed processing algorithms can be used in a parallel way by using Spark. For this purpose, the platform user is provided with a Spark-based orchestrator, which takes advantages of the Spark parallel framework. It enables different data entries to be processed simultaneously by a given module, which has been previously deployed in several machines of the cluster. The orchestrator distributes the processing load between the different instances of the modules. In addition, in the case of those modules or functionalities directly implemented in Spark (i.e., modules that have been written according to the map-reduce paradigm), data are optimally processed in parallel.
- Distributed storage. In order to accelerate the modules in the MixedEmotions platform, a distributed storage system has been proposed. This approach provides the platform with the scalability required to manage large amounts of data. Furthermore, distributed storage is aligned with parallel processing, since data can be simultaneously read/write from several nodes in the cluster. Specifically, HDFS and Elasticsearch have been chosen for the implementation of distributed data storage.

4. Strategies for data processing acceleration

In this section, we describe the main strategies adopted for the acceleration of the algorithms for data analysis (i.e., entity, topic, sentiment, emotion, suggestion,...) in the MixedEmotions platform. Once the initial analysis of the algorithms was completed, those identified as distributable modules were selected to be accelerated.
As a common strategy to ensure the interoperability of the modules and improve its efficiency, we have defined a common format to represent the information in the platform. Additionally, we have considered two approaches to improve the performance of the processing algorithms: the implementation in a parallel framework like Spark and the parallelization of standalone modules. These are described in the following points.

4.1. Common strategies

As an initial step to ensure the optimal performance of the modules composing the MixedEmotions platform, we have defined the format that they must adopt to exchange information with other modules or elements of the platform. To accommodate both structured and semi-structured data, JSON was chosen as the most suitable information format.

The JSON format defined for MixedEmotions corresponds to a special case known as JSON-LD (JSON for Linked Data). It is a version of the well-known JSON approach adapted to linked data. Hence, it enables to format data entries that are linked.

In MixedEmotions, we have developed specific tools for the platform capable of receiving a common string as input while yielding the corresponding JSON-LD.

4.2. Implementation of processing modules in Spark

The most efficient way to accelerate processing algorithms is to program them according to the properties of the framework adopted for parallel data processing. In MixedEmotions, this framework is Spark, which enables efficient parallel data processing based on in-memory storage. Spark defines the Resilient Distributed Dataset (RDD), which is an abstraction to refer to a collection of elements, all of them of the same type, distributed among the nodes of the cluster.

Programming an algorithm in Spark involves handling RDDs. For instance, in MixedEmotions, the RDD may contain tweets from which we expect to evaluate the sentiments or emotions expressed by a Twitter user. For the development of Spark programs, the distributed nature of the RDDs must be taken into account. For instance, it must be appreciated that the elements of the RDD are stored in different physical machines and there is no order for these elements. Therefore, transformations of these elements must be equally applied to each of them. Similarly, computations on the elements of the RDD must involve associative and commutative operators.

To accelerate them, we have implemented some of the NLP algorithms taking into account the properties of Spark. Specifically, the algorithms accelerated correspond to entity and topic extraction from text in Spanish:

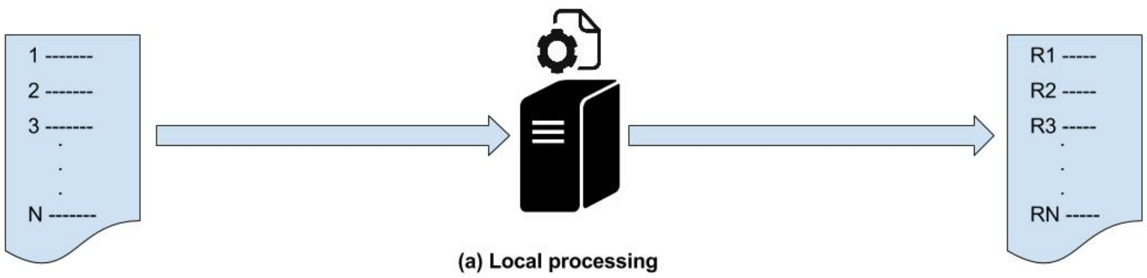
- Entity extraction aims to identify the subjects and individuals mentioned in the text. It tries to answer the question "Who is the text talking about?" To this end, a dictionary is taken as reference, which contains a list of entities together with the number of Wikipedia entries where they appear. If one of them is found in the underlying text, it is considered a valid entity if the number of associated links is higher than a predefined threshold. The process runs in a parallel way by loading the dictionary file and distributing its content as an RDD. It is processed together with the RDD of input entries, each of them being a String according to a JSON format.
- Topic extraction aims to identify the main domains or areas referred to by the text. Specifically, topic extraction answers the question "What is the text talking about?". It is performed using a taxonomy that associates words with topics. For instance, it links the word "árbitro (referee)" with the topic "fútbol (football)" or the word "investigación (research)" with the topic "innovación (innovation)". As for the entity extraction algorithm, the implementation of the topic extractor in Spark involves the parallelization of the taxonomy. To this end, its content was parsed and distributed as an RDD. Similarly, the set of entries to be processed were included in an RDD, which was combined with the taxonomy in the parallel framework.

For both modules, preliminary versions were initially available as a Python program. It must be appreciated that the specific approach for text analysis implemented in these versions was not modified when the modules were programmed in Spark. Thus, the algorithm provides the same result when either the Python or the Spark implementation is used. As can be observed, both algorithms are based on the use of taxonomies or dictionaries. For the implementation of the modules in Spark, the content of these resources is distributed by using a RDD.

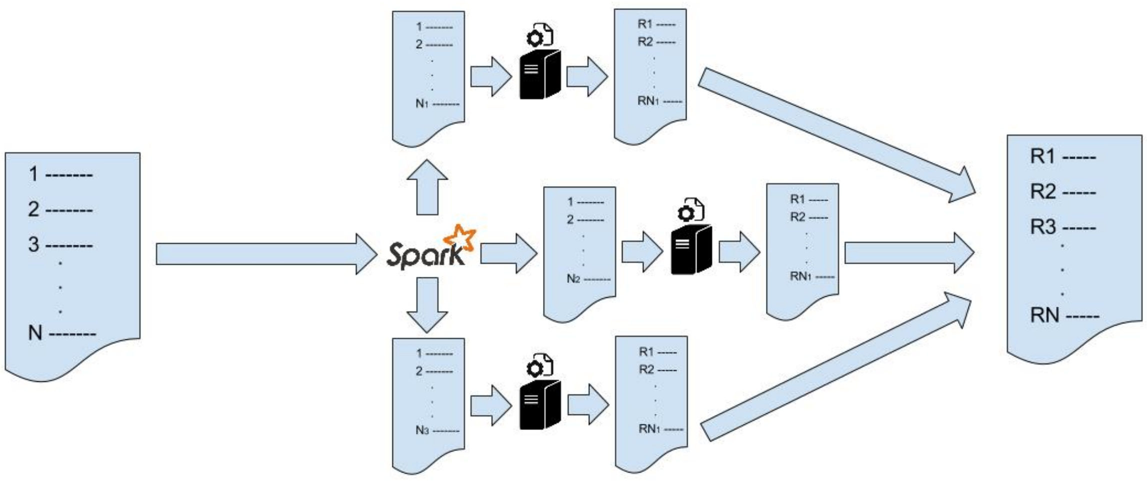
4.3. Parallelization of processing modules

The second approach for the acceleration of the algorithms is based on the use of a Spark-based wrapper for the underlying module. In this approach, the module (processing functionality) behaves as a local application. The software was not developed taking into account the attributes of a parallel processing framework. Thus, it processes each entry sequentially instead of computing results for several entries simultaneously. The following strategy is then used to obtain a parallel scenario. First, the module (local program) is installed in each machine of the cluster. Hence, the strategy consists of distributing the data between these machines, which will process the entries simultaneously. As a result, each copy of the module works locally in each machine but, globally, a parallel process is obtained.

The key point in this accelerating strategy is the software responsible for distributing the original collection of data entries between the cluster machines. This task is allocated to Spark, the distributed processing framework adopted in MixedEmotions. Therefore, a wrapper is required to simulate the behaviour of a Spark module. This wrapper would accept an RDD of entries as input, returning the corresponding RDD of results. In the body of the wrapper, each partition of the RDD is simultaneously processed by the machines of the cluster. The following figure shows a scheme for the comparison between local (sequential) and distributed (parallel) processing of the data.



(a) Local processing



(b) Distributed processing

5. Conclusions

This deliverable addressed the initial strategies and actions taken in regard of the optimization of the software modules that compose the MixedEmotions platform. In this context, the term optimization refers to the integration of the module in the platform (i.e., to make it compatible with the rest of the components) and the improvement of its efficiency (i.e., to maximise the number of items the module is capable of processing either by programming it according to a map-reduce style or by parallelizing the computation). To facilitate the integration and compatibility of a module with the rest of the platform elements, the JSON-LD format has been proposed as the protocol to manipulate information. In addition, in order to implement parallelization strategies, a Spark orchestrator has been proposed. It is in charge of distributing data among the cluster machines, as well as of the division of large tasks into smaller ones.

The trials carried out to validate these approaches show a trade-off between the improvement derived from parallelization and the cost of its use. As commented above, Spark was used as the distributed processing framework. Its use contributes to increase the complexity of the software. Furthermore, no improvement on the latency could be observed in case a reduced amount of data is to be processed. On the other hand, parallelization enables the scalability of the platform, while reducing the latency if the processing task must deal with large volumes of data. As a result, the parallel approach is recommended for use by the MixedEmotions platform.