MULTISENSOR

Mining and Understanding of multilinguaL contenT for Intelligent Sentiment Enriched coNtext and Social Oriented inteRpretation

FP7-610411

D7.5

Crawling infrastructure

Dissemination level:	Public
Contractual date of delivery:	Month 20, 30/06/2015
Actual date of delivery:	Month 20, 30/06/2015
Workpackage:	WP7 System Development and Integration
Task:	T7.2 Crawlers and data channels infrastructure
Type:	Prototype
Approval Status:	Draft
Version:	1.0
Number of pages:	29
Filename:	D7.5_CrawlingInfrastructure_2015-06-30_v1.0.pdf

Abstract

This document describes the crawling infrastructure developed in Task 7.2. More specifically, it reports the techniques and the freely available crawlers used in the MULTISENSOR platform.

co-funded by the European Union

History

Version	Date	Reason	Revised by
0.1	05/05/2015	Initial table of contents	I. Arapakis (BM-Y!)
0.2	15/05/2015	Final table of contents	I. Arapakis, (BM-Y!), S. Vrochidis (CERTH)
0.3	15/06/2015	Contributions	B. Cambazoglu (BM-Y!), T. Forellat, M. Puigbo (PIMEC), E. Jamin (EVERIS), L. Blacha (PR), B. Simeonov (ONTO)
0.4	20/06/2015	Integrated document	I. Arapakis (BM-Y!)
0.5	25/06/2015	Review	C. Doulaverakis (CERTH)
1.0	30/06/2015	Final document	I. Arapakis (BM-Y!)

Author list

Organization	Name	Contact Information
BM-Y!	Ioannis Arapakis	arapakis@yahoo-inc.com
BM-Y!	B. Berkant Cambazoglu	barla@yahoo-inc.com
PIMEC	Teresa Forrellat	tforrellat@pimec.org
PIMEC	Marti Puigbo	mpuigbo@pimec.org
EVERIS	Emmanuel Jamin	
ONTO	Boyan Simeonov	
PR	Leszek Blacha	
CERTH	Stefanos Vrochidis	stefanos@iti.gr

Executive Summary

In this deliverable we present the crawling infrastructure of MULTISENSOR. To deal with the large amount of web data and data originating from social media platforms, we deployed a suite of web crawlers, social media data collectors, and API wrappers that use filters and seed lists to collect only relevant information. More specifically, three sources of data are crawled: real-time sociometric counts from social media platforms, Twitter streaming data, and web pages. We present an architecture that crawls the above types of sources, processes the content (e.g., removal of boilerplate), and stores all data into a central repository for further processing by the MULTISENSOR services. Specifically the crawling architecture consists of three main components. A collector for supporting real-time aggregation of sociometric counts, a crawler for crawling web pages using a seed list and API wrappers for retrieving structured data from knowledge bases. The web crawler is based on the large scale Nutch open source crawler and is used both for focused crawling of web pages as well as for crawling multimedia objects. The crawlers run on a server and are based on the Apache Hadoop open framework.

The deliverable provides information on the open source crawlers employed, it describe the access to APIs of web crawlers for media article collection and finally it provides access to the code of the sociomertric collector.

Abbreviations and Acronyms

API	Application Programming Interface
CMR	Central Multimedia Repository
CNR	Central News Repository
DB	DataBase
HDFS	Hadoop Distributed File System
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
PR	PressRelations
RDF	Resource Definition Framework
SSH	Secure Shell
UC	Use Case
UCS	Universal Character Set
URL	Uniform Resource Locator
UTF	UCS Transformation Format
W3C	World Wide Web Consortium
XLS	eXceL Spreadsheet
XML	eXtensible Markup Language

Table of Contents

1 INTRODUCTION

Crawling and analyzing web data and data coming from social media sites at large scale is very challenging even for a system with a very large amount of h/w resources. The huge volumes of information and the abundance of noise demand for intelligent crawling techniques. In MULTISENSOR we develop a suite of crawling tools that can focus around a given set of web domains and bring relevant information from the crawled pages. Specifically the crawled information is used for populating the repositories of MULTISENSOR with content (e.g. web articles, social media), as well as with information that is required for the indicators for SME internationalization (WP3). In this context this deliverable presents the crawling infrastructure developed in MULTISENSOR.

Figure 1 below describes the overall crawling architecture of MULTISENSOR. The crawler component of the architecture is responsible for periodically feeding the Central News Repository with news related to the use cases (see D7.4, section 2.3.2.1). The crawler aggregates news from several heterogeneous sources into a common, JSON-based format in the news repository. It is scheduled to run daily, gathering news from all the configured sources and updating the repository. Once the collected data is stored it is then consumed by the rest of the MULTISENSOR pipeline services, such as speech analysis, NE recognition, and concept extraction (WP2), content extraction, sentiment analysis, and social media mining (WP3), topic detection, and multimodal indexing and retrieval (WP4) and summarization (WP6).

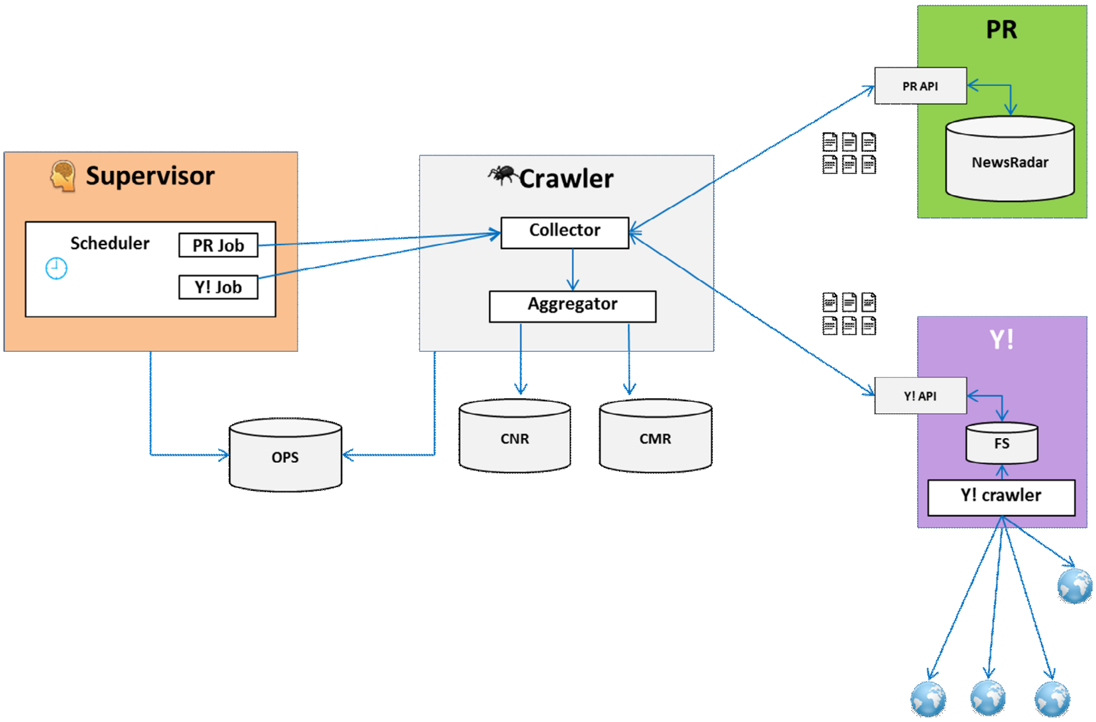

Figure 1: Web crawling architecture

When considering social media sources, we distinguish between three main types of data sources: sociometric counts which typically come as time series (streams) of high volume, data from structured knowledge bases, and unstructured, heterogeneous data that is crawled from the web. Those three types of sources work in synergy and are complementary. The sociometric counts can be helpful in monitoring the evolution of a news article's popularity online and can be augmented with data that comes from web sources, which is much more descriptive and detailed.

In MULTISENSOR we track those two sources of data. For the real-time sources we develop a stream processing architecture and store the data as time series. For the web sources such as news agencies, we use the URLs in predefined seed lists to start web crawling and focus the crawling on URLs that match the web domains of interest. Typically, crawling consists of fetching a page, extracting the hyperlinks in the page, and then systematically fetching all of those pages that are hyperlinked. This process is repeated to an arbitrary depth, depending on our objective. The basic algorithm for a web crawl can be framed as a *breadth-first search*, which is a fundamental technique for exploring a space that's typically modeled as a tree or a graph given a starting node and no other known information except a set of possibilities. In our web crawl approach, our starting node would be the initial web page from the seed list and the set of neighboring nodes would be the other pages that are hyperlinked.

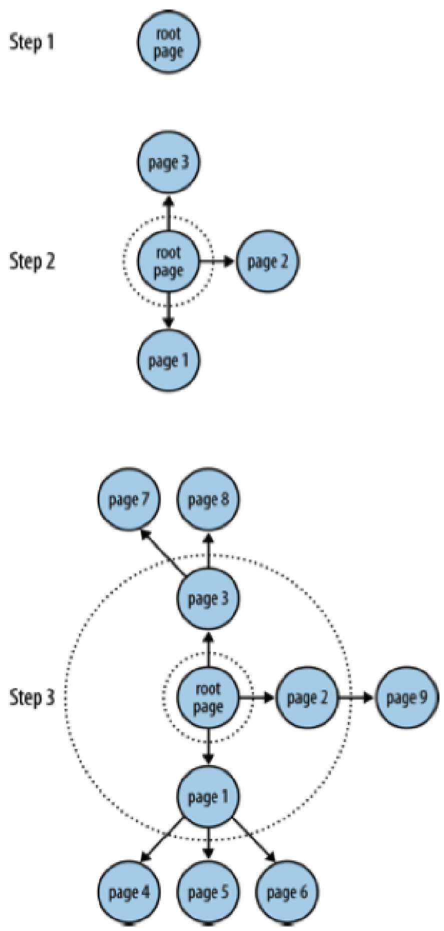

Figure 2: Breadth-first search where each step of the search expands the depth by one level until a maximum depth or some other termination criterion is reached

Standard performance analysis of any algorithm generally involves examining its worst-case time and space complexity - in other words, the amount of time it would take the program to execute, and the amount of memory required for execution over a very large data set. The breadth-first approach we employ to frame a web crawl is essentially a breadth-first search, except that we're not actually searching for anything in particular because there are no exit criteria beyond expanding the graph out either to a maximum depth or until we run out of nodes. For a breadth-first search (or breadth-first crawl), both the time and space complexity can be bounded in the worst case by b^d, where b is the branching factor of the graph and d is the depth (Figure 2). Despite the fact that our crawler is focused on specific web domains, still those domains can generate a large amount of data, so we need to have a design that is scalable and efficient. In order to achieve this the web crawler is developed on top of the distributed infrastructure Hadoop.

The rest of this report as organized as follows. We start in Section 2 with a description of the architecture of the online news crawlers, followed by a presentation of the social media collectors and data wrappers in Section 3. Then, in Section 4, we discuss the retrieval of data from structured knowledge bases and in Section 5 we describe the crawler storage infrastructure. We finish in Section 6 with a summary and conclusions.

2 ONLINE NEWS CRAWLERS

2.1 BM-Y! Web Crawler

Discovering and downloading newly published articles in news web sites in a timely manner requires implementing a web crawling architecture that involves a number of components. Developing a crawler from scratch is a non-trivial task due to the difficulties involved in implementing a robust and scalable crawler that can cope with the chaos and malicious intent in the Web. Therefore, we opt for a modular crawling architecture in which existing software components, whose robustness and efficiency were previously proven, are combined in a meaningful manner for the discovery and download of news articles. Our architecture combines various big data processing technologies, such as Hadoop, HDFS, and HBase. Hence, it offers efficient and scalable web crawling functionality.

In the following sections, we provide an overview of the functionality of each component in our architecture. We then describe the workflow of the crawler. Next, we explain the details of the script we implemented for automated installation and configuration of the crawler. Finally, we explain the format of the input and output files generated by the crawler with examples.

2.1.1 Crawling architecture

The crawling component, which will discover and fetch news pages, lies at the heart of our architecture. This component is responsible for three main tasks. First, it reads URLs from a priority queue and downloads the corresponding web pages from the Web. Second, it parses the content of downloaded pages to discover new URLs. Third, it stores the web pages in a data store. In our architecture, as the crawling component, we use Apache Nutch[1], which is an extensible and open source web crawler project. It has been used many times in other projects with success. Nutch supports distributed crawling, and it can thus scale to crawl millions of web pages. Crawling is performed through MapReduce[2] jobs running on the Apache Hadoop[3] platform.

Hadoop is a software library and a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models, such as MapReduce. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. The library is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures. Therefore, Hadoop is a perfect fit for the web crawling task, which is a long-lasting task that can benefit from the provided fault tolerance.

Nutch needs a storage system to store two types of output: extracted links and downloaded pages. The extracted links can be stored on a temporary storage, as they are used to discover new pages and are then discarded. To this end, we use HDFS, which is

[1] http://nutch.apache.org/

[2] http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

[3] http://hadoop.apache.org/

the default storage system in Hadoop. HDFS provides good performance for bulk reading of the data, but it does not provide random access. Therefore, for the downloaded pages, we use a separate data store, Apache HBase[4]. HBase supports random, realtime read/write access to Big Data (e.g., billions of records with millions of fields) on clusters of commodity hardware. It is an open-source, distributed, versioned, non-relational database modelled after Google's Bigtable. Apache HBase provides Bigtable-like data access and update capabilities on top of Hadoop and HDFS.

2.1.2 Crawling workflow

We perform crawling in batch mode. That is, the crawling process is performed in sessions and is not continuous. In each session, the crawling process begins with a set of seed pages and stops after a certain constraint is satisfied. For example, the crawler may stop after a certain number of pages are downloaded or the size of the news page repository reaches a certain limit. In our case, the crawling process stops as soon as all web pages within a certain link hop distance (with respect to the seed pages) are downloaded. By increasing this distance threshold, we can adjust the coverage of the crawler. A new crawling session can be started periodically, e.g., every day at midnight. The pages crawled in the previous session are discarded before the news sessions starts.

Each crawling session starts by reading a number of seed URLs stored in a text file on HDFS. The seed URL file provides the entry URLs for different news web sites whose content is of interest to the crawler. The content of this file is created and updated manually.

A typical crawler works as follows. The seed URLs are first placed in a download queue, which is implemented as a FIFO priority queue. The crawler then iterates until the priority queue becomes empty or the above-mentioned stopping condition is met. At each iteration, the crawler removes the URL at the head of the priority queue. A fetched thread contacts the web server hosting this URL. Once the content of the URL is retrieved from the web server, it is passed to the parsing module. This module is responsible for extracting the links within the downloaded web page. The extracted links are canonicalised[5] and a URL-seen test is performed for each link. This test checks for the existence of the extracted URLs among the previously seen URLs. If the URLs are seen for the first time, they are added to the tail of the download queue. Otherwise, they are discarded to prevent duplicate downloading of the same page. The download HTML page is passed to the storage manager, which stores the page together with some additional meta-data. The crawling process continues with the next page in the download queue.

In Nutch, the above-mentioned process is implemented as a MapReduce job, which has two main steps. In the first step, the URLs in the priority queue are fetched from the Web and stored temporarily on HDFS. In the second step, the links are extracted and added to the priority queue. This two-step MapReduce job is executed multiple times until the crawling process finishes. Each execution of the MapReduce job fetches web pages that are of certain hop distance from the initial seed URLs. Therefore, the entire process results in a breadth-first crawling of the Web.

[4] http://hbase.apache.org/

[5] https://en.wikipedia.org/wiki/Canonical_link_element

In our architecture, the downloaded pages are stored in HBase. After the crawling session is completed, a small Java code is executed to read all web pages and meta-data from HBase. This code writes the final output of the crawler into a text file on the local storage in JSON format. The entire crawling architecture is illustrated in Figure 3.

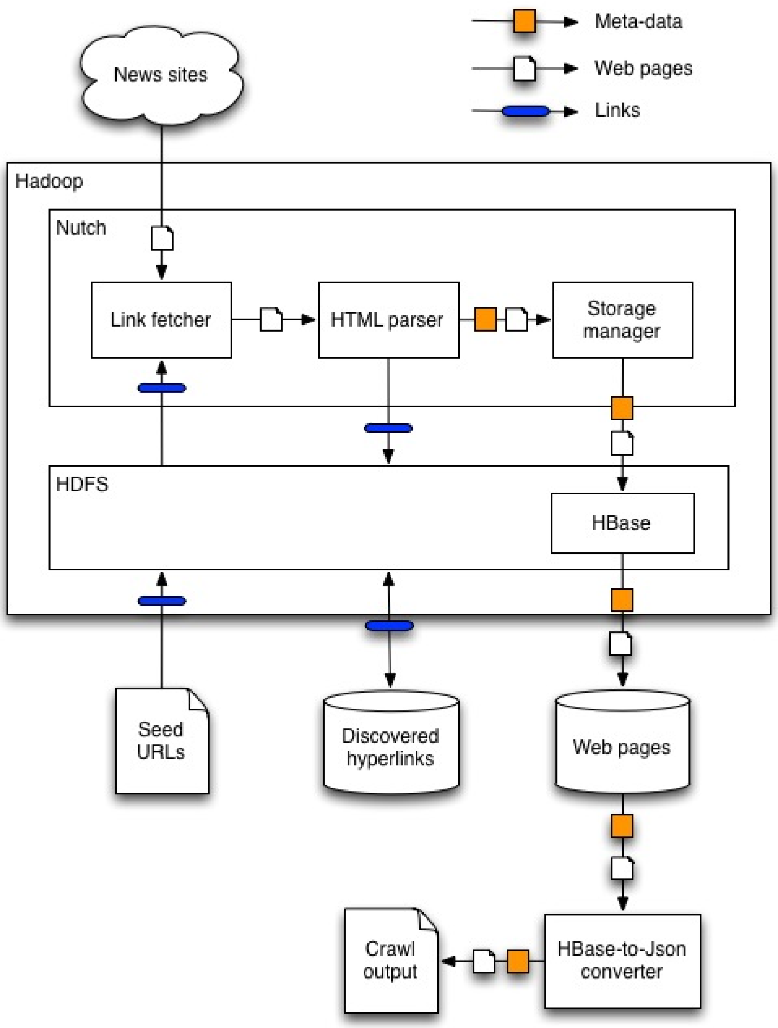

Figure 3: Crawling architecture of Yahoo crawler.

2.1.3 Installation, configuration, and execution

We automated all tasks related to installation, configuration, and execution of the components mentioned in the previous sections. Moreover, these tasks can be all carried out remotely through SSH. To this end, we used the Fabric[6] library in Python. This is a command-line tool for streamlining the use of SSH for application deployment or systems administration tasks. We implemented a simple fabric script (fabfile.py) that performs the tasks described in Table 1.

Step	Command	Description
Installation	fab -f fabfile.py setupCluster	This command is used at the very beginning to download the installation

[6] http://www.fabfile.org/

		files for Hadoop, HBase, and Nutch, and Solr (we install Solr because of the dependency of Nutch to it). The downloaded pages are stored on the master node, where we will perform the installation. All files are simply extracted under a root directory on the local file system. Moreover, certain configuration is performed. For example, for Hadoop, we add to a config file the IP addressed of nodes that will constitute the master and slave nodes in the Hadoop system. For Nutch, we customize certain crawling parameters.
Starting	fab -f fabfile.py startCluster	This command is used to start different daemons on the master node. In particular, we start all daemons required by Hadoop, HBase, and Solr using this command. The file that keeps the seed URLs are also copied from the local file system to HDFS when this command is executed.
Stopping	fab -f fabfile.py stopCluster	This command is used to stop the running daemons. It has to be executed before making any modifications in the config files.
Cleaning	fab -f fabfile.py refreshCluster	This command is used to delete all temporary files created on HDFS and HBase. Both are returned to a clean, initial state.
Crawling	fab -f fabfile.py runNutch	This is the main command to execute the crawler. It accepts four parameters: The location (on HDFS) of the seed URL file, the name of the database to be created on HBase, the URL of the Solr system, and finally, the depth of the crawl.

Table 1: Set of actions applied by the fabric script

A typical execution sequence for these commands would be as follows:

```
// fresh installation and configuration
fab -f fabfile.py setupCluster

<new crawling session starts>

// clean the temporary files
fab -f fabfile.py refreshCluster
```

```
// start the daemons and copy the seed URLs to HDFS
fab -f fabfile.py startCluster

// start the crawler
fab -f fabfile.py runNutch

// convert the crawl data in HBase to a Json file (using a Java code)
// stop the daemons
fab -f fabfile.py stopCluster

<crawling session ends>
```

2.1.4 Input and output formats

The only input file for the crawler is a text file that keeps the URLs used as seeds. The format of this file is very simple. In each row of the file, we have a separate URL listed. At the moment, we use a seed file with 562 URLs. The first few lines of our seed file is shown below:

```
// Seed list
aon.at
apa.at
atv.at/Channel.aspx
austriainnovativ.at
besser-wohnen.co.at
brancheintern.at
burgenland.orf.at
…
```

The final output of the crawler is a text file that contains the content of the pages downloaded in the last crawling session, together with some meta-data. The information is encoded in JSON format. In particular, the final output includes the following fields for every web page:

- **url:** URL of the page
- **c_sourcecode:** HTML content of the page
- **crawled:** The timestamp (YYYY-MM-DD) indicating the time the page was downloaded
- **title:** Title of the page
- **body:** Body content of the page (after removing the HTML tags)

Below is an example output including two web pages:

```
{
  "webpages" : [ {
     "body" : "Новини / Споделете всичко за вашите любими звезди и
научете …",
     "c_sourcecode" : "<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0
…",
     "country" : "bg",
     "crawled" : "2015-8-21",
     "language" : "bg",
     "source" : "www.bliasak.bg",
     "title" : "Новини / Споделете всичко за вашите любими звезди и
научете ...",
     "url" : "http://www.bliasak.bg/fanclub/news/",
     "_analyzer" : "bulgarian"
  }, {
```

```
    "body" : "Конкурси / Споделете всичко за вашите любими звезди и научете …",
    "c_sourcecode" : "<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0 …",
    "country" : "bg",
    "crawled" : 1436198900900,
    "language" : "bg",
    "source" : "www.bliasak.bg",
    "title" : "Конкурси / Споделете всичко за вашите любими звезди и научете ...",
    "url" : "http://www.bliasak.bg/fanclub/p2_814_0.html",
    "_analyzer" : "bulgarian"
  } ]
}
```


2.2 PR Web Crawler

Pressrelations aggregates news from media all over the world via proprietary crawling technologies and provides access to news items that have been preselected for topics relevant to the MULTISENSOR project (i.e. referring to the defined use cases). News sites are crawled for selected keywords that are relevant to the three use cases. The keyword list has been created by the use case partners (see example bellow).

```
// Keyword list
66758
YOGURT*
66758
YOGUR*
66758
YOGHURT*
66758
YAOURT*
66758
MUELLER +YAOURT*
66758
MUELLER +JOGHURT*
66758
EHRMANN +MILCH*
66758
EHRMANN +YOGURT*
66758
EHRMANN +YOGUR*
…
```

The Media Collector accesses the JSON-based-API at pressrelations and retrieves content incrementally as new articles are available.

2.2.1 Multimedia dimension

For radio and TV articles no audio and video material is provided other than textual information. For internet articles, links to multimedia material (images, videos, audio) is provided, if they can be extracted by the PR crawlers. However, this is observed on average only in a few cases due to the heterogeneous structure of the news sites that are being crawled. Table 2 shows the type and estimated monthly volumes of the crawled content.

Type	Volume

Internet items (e.g., news sites, social media)	~ 110.000
Agency articles	~ 1-10
Radio and TV articles (only up to September 2014)	~ 1-10
Printed articles	~ 100

Table 2: Type and estimated monthly volumes of crawled content

2.2.2 Technical description

Access to the pressrelations API to retrieve crawled news items

i. Internet articles

- Resource URL: http://multisensor.pressrelations.de/api/news/internet/
- Mandatory parameters:
  - customerKey - Hexadecimal, the internal ID of the customer at PR
- Optional parameters:
  - filterSources - String, internet,socialmedia (default = "") (You can filter if you want to get news only from online news websites or social media channels.)
  - count - Integer, 1 - 5000 (default = 10)
  - maxId - Integer, "article_id" from the articles retrieved (If maxId is included in the URL then all articles older then the maxId will be displayed. The article with this maxId will also be displayed.)
  - sinceId - Integer, "article_id" from the articles retrieved (If sinceId is included in the URL then all articles newer than the sinceId will be displayed. The article with the sinceId will NOT be displayed.)

ii. Print, radio, TV and internet articles

- Resource URL: http://multisensor.pressrelations.de/api/news/press/’
- Mandatory parameters:
  - customerKey - Hexadecimal, the internal ID of the customer
- Optional parameters:
  - filterSources - String, radio,press,tv,agency (default = "") (You can filter if you want to get news only from press, radio, tv or agency channels.)
  - count - Integer, 1 - 5000 (default = 10)
  - maxId - Integer, "article_id" from the articles retrieved (If maxId is included in the URL then all articles older then the maxId will be displayed. The article with this maxId will also be displayed.)
  - sinceId - Integer, "article_id" from the articles retrieved (If sinceId is included in the URL then all articles newer than the sinceId will be displayed. The article with the sinceId will NOT be displayed.)

2.2.3 Output of JSON API

An example of JSON response from the pressrelations API is shown bellow:


```
{
  "results": [
     {
```

```
      "use_case" : "UC1 - Household Appliances",
      "language" : "en",
      "country" : "INT",
      "_id" : "",
      "crawled" : "2014-06-17",
      "multimediaUrls" : [],
      "c_sourcecode" : "<HTML>original source code</HTML>",
      "source" : "facebook.com",
      "pr_summary" : "",
      "date_timestamp" : 1402992610,
      "body" : "Get the most evenly cooked food with Whirlpool's 6th
sense cooking technology!\n\nHere's how it works.\n\nVideo courtesy
Whirlpool Europe",
      "feed" : "",
      "date" : "2014-06-17",
      "article_id" : 1372557200,
      "pr_feed" : "internet",
      "url" :
"http://www.facebook.com/permalink.php?story_fbid=245151655673882&id=1
29981993857516",
      "title" : "Video: Whirlpool India"
    },
    {
      "use_case" : "UC1 - Energy Policy",
      "language" : "de",
      "country" : "DE",
      "_id" : "",
      "crawled" : "2014-06-16",
       "multimediaUrls" : [
                "http://images.zeit.de/wirtschaft/2014-
03/gas_russland/gas_russland-220x124.jpg",
                "http://images.zeit.de/wirtschaft/2014-
06/biblis/biblis-220x124.jpg",
                "http://images.zeit.de/wirtschaft/2014-
06/europaeische-zentralbank-draghi/europaeische-zentralbank-draghi-
220x124.jpg",
                "http://images.zeit.de/wirtschaft/2014-06/gazprom-
gas.ukraine-eu/gazprom-gas.ukraine-eu-540x304.jpg",
                "http://images.zeit.de/wirtschaft/2014-06/pakete-
onlineshopping/pakete-onlineshopping-220x124.jpg",
                "http://images.zeit.de/wirtschaft/2014-
06/umverteilung/umverteilung-220x124.jpg",
                "http://images.zeit.de/zeit-verlag/2010-
05/handelsblatt-teaser/handelsblatt-teaser-feedicon.jpg"
      ],
      c_sourcecode : "",
      "source" : "Die Zeit",
      "pr_summary" : "Article summary written by PR",
      "date_timestamp" : 1402920008,
      "body" : "Ukraine-Krise: Russland stoppt Gaslieferungen an
Ukraine\r\n\r\nRussland hat nach Angaben aus Kiew seine Gaslieferungen
an die Ukraine eingestellt. Die Regierung in Kiew versichert, die
Versorgung Europas sei aber garantiert.\r\n16. Juni 2014 11:59
Uhr\r\n\r\nArbeiter an einer Gaspipeline in der Ukraine © Gleb
Garanich/Reuters\r\n\r\nRussland hat nach Angaben aus Kiew seine
Gaslieferungen an die Ukraine eingestellt. Die ukrainische Regierung
sei darüber informiert worden, dass die Gaslieferungen \"auf Null\"
heruntergefahren worden seien, sagte der Energieminister Jurij Prodan,
nachdem die letzte Verhandlungsrunde im Gasstreit gescheitert war.
Allerdings werde die Ukraine sicherstellen, dass für Europa bestimmtes
Gas in die entsprechenden Länder weitergeleitet werde, sagte
```

```
er.\r\n\r\nDas ukrainische Energieunternehmen Naftogaz erklärte, das
Land habe so viel Gas in den Speichern, dass die Versorgung bis
Dezember sichergestellt sei. Die Bundesregierung befürchtet für
Deutschland keine Gas-Lieferengpässe. \"Eine Gefährdung der
Versorgungssicherheit in Deutschland können wir auch durch die neue
Entwicklung nicht erkennen\", sagt ein Sprecher des
Wirtschaftsministeriums. Transitlieferungen durch ukrainisches Gebiet
seien nicht betroffen.\r\n\r\nAnzeige\r\n\r\nIn der Nacht zum Montag
waren Verhandlungen zwischen Russland und Ukraine über die Begleichung
von Schulden sowie den künftigen Gaspreis gescheitert. Der
Staatskonzern Gazprom besteht deswegen auf Vorkasse. Die Beziehungen
zwischen beiden Staaten sind zudem angespannt, weil prorussische
Separatisten in der Ostukraine für eine Abspaltung der Region
kämpfen.\r\n\r\nDie russische Regierung zeigte sich aber prinzipiell
zu weiteren Gas-Verhandlungen mit der Ukraine bereit.
Ministerpräsident Dmitri Medwedew nannte als Bedingung, dass das
Nachbarland die aufgelaufenen Schulden vollständig
begleicht.\r\nKlagen vor Schiedsgericht\r\n\r\nBei Länder reichten
wegen des Gasstreits Klagen bei der internationalen Schiedsstelle für
Handelsstreitigkeiten in Stockholm ein. Der russische Staatskonzern
Gazprom klagt einer Mitteilung zufolge wegen ukrainischer Schulden für
nicht bezahlte Gaslieferungen von 4,458 Milliarden US-Dollar (3,290
Milliarden Euro). Der ukrainische Energieversorger Naftogaz reichte
hingegen eine Klage gegen Gazprom ein wegen zu hoher Preise von
aktuell 485,5 US-Dollar je 1.000 Kubikmeter Gas.\r\n\r\nDie
Schiedsstelle in Stockholm - Arbitration Institute - solle einen
Marktpreis für russisches Gas festlegen, teilte Naftogaz in Kiew mit.
Demnach verlangt die Ukraine von Russland auch sechs Milliarden US-
Dollar Rückzahlung für überteuerte Gaslieferungen. Kiew und Moskau
hatten sich zuvor unter Vermittlung von EU-Kommissar Günther Oettinger
nicht auf einen neuen Gaspreis einigen können.\r\n\r\n@",
      "feed" : "",
      "date" : "2014-06-16",
      "article_id" : 1371797639,
      "pr_feed" : "internet",
      "url" : "http://www.zeit.de/wirtschaft/2014-06/ukraine-russland-
gasstreit-lieferstopp/komplettansicht",
      "title" : "Ukraine-Krise Russland stoppt Gaslieferungen an
Ukraine (11:59, ZEIT ONLINE)"
    }
  ],
  "errors" : [

  ],
  "nextPageUrl" :
"/api/news/internet?count=10&maxId=1372650042&customerKey=?????"
}
```

3 SOCIAL MEDIA CHANNELS

3.1 BM-Y! Sociometrics Collector

A sociometrics collector[7] has been implemented, which includes a collection of wrappers for the following social media APIs:

i. Twitter
ii. Facebook
iii. Google Plus
iv. LinkedIn
v. Pinterest
vi. Delicious
vii. StumbleUpon
viii. Reddit
ix. Digg

Most of these social media platforms offer robust and well-documented gateways into a very comprehensive and well-organized information store, both in terms of breadth and depth. It's broad in that its user base represents about one-seventh of the entire living population, and it's deep with respect to the amount of information that's known about any one of its particular users.

The collector applies a multi-threaded approach, meaning that for a given URL (associated with a unique id) it performs concurrent calls to the APIs of the above platforms and returns the results in JSON format. A *settings.properties* file allow to configure the API parameters that are specific to each social media platform, as well more generic settings such as the type of *request method*, the *connect timeout*, and the *read timeout*. The sociometrics collector service is run as a deployed web service on the Grinder server and allows the following parameters:

Parameter	Default	Description
url (required)	none	The URL of the page we want to fetch the sociometric counts for
Id (required)	None	A unique id of the page we want to fetch the sociometric counts for
format (fixed)	json	Currently the service supports only json format
Callback	Fetcher()	The Java function to execute

Table 3: The sociometrics collector service parameters

A typical input/output of the sociometrics collector is shown bellow. Basically, the output is a collection of sociometric counts, which can be stored and analyzed as time series. More specifically, this data can be used to perform trend analysis, i.e., analyse time-

[7] Available for download at: https://quark.everis.com/svn/MULTISENSOR/trunk/wp7/ms-crawler-socialmedia/

varying data to identify the trends or to predict the future outcome of a target variable. Another very popular task, which has been investigated in different contexts, is the prediction of an item's popularity over time.

```
// given a news article with a unique ID and URL
id = "00001"; url = "http://www.linkedin.com";

// the service retains a crawling cycle timestamp (generated once it's
first instantiated), and a timestamp for each URL connection made
{
     "twitter" : [
            {
                   "tweet_count" : 136789
            }
     ],
     "linkedIn" : [
            {
                   "share_count" : 152542
            }
     ],
     "_crawl_cycle_timestamp" : 1393607209433,
     "_id" : "12345",
     "_url" : "http://www.linkedin.com",
     "reddit" : [
            {
                   "comments_count" : 0,
                   "downs_count" : 0,
                   "ups_count" : 1,
                   "score_count" : 1
            }
     ],
     "facebook" : [
            {
                   "comment_count" : 1968,
                   "like_count" : 4065,
                   "share_count" : 17263
            }
     ],
     "_url_timestamp" : 1393607209634,
     "stumbleUpon" : [
            {
                   "views_count" : 7089
            }
     ],
     "delicious" : [
            {
                   "bookmarks_count" : 11944
            }
     ],
     "pinterest" : [
            {
                   "shares_count" : 37
            }
     ],
     "googlePlus" : [
            {
                   "shares_count" : 316424
            }
     ]
}
```

The related work on this problem includes popularity prediction of multimedia content (Pinto et al., 2013; Shamma et al., 2011), social marketing and stock market prediction (Yu et al., 2011; Zhang et al., 2011), election prediction (Tumasjan et al., 2010), impact prediction of research articles (Brody et al., 2006), topic volume prediction (Lehmann et al., 2012; Ruan et al., 2012), or early detection of popular online content in social media (Kim et al., 2011; Mathioudakis et al., 2010).

There is also a fairly large number of studies on popularity prediction in the context of online news (Ahmed et al., 2013; Freyne et al., 2010; Garimella and Castillo, 2014; Gupta et al., 2012; Jamali and Rangwala, 2009; Lerman and Hogg, 2010; Marujo et al., 2011; Szabo and Huberman, 2010; Tatar et al., 2011; Tsagkias et al., 2010).

3.2 PR Social Media Data Wrappers

Pressrelations is currently able to crawl and deliver posts from the following social media channels:

i. Twitter
ii. Facebook
iii. YouTube
iv. Blogs
v. Forums
vi. Consumer Portals
vii. Wikipedia

Depending on the channel, either the API of the source is crawled (e.g., Facebook, Twitter, YouTube) or the related sources (e.g., a list of Consumer Portals). The results of search engines, directories and manual desk research are also included to extend our source pool.

The media data collected are determined by the source. If we crawl an API we are able to track all available information that is connected with a single post: Social media articles (tweets, posts etc.) as well as articles from internet newspapers/magazines can be retrieved via the pressrelations API as described in section 2.2. When calling the pressrelations API, the optional parameter “filterSources = socialmedia” can be added to the URL to retrieve articles from social media channels only.

The number of posts crawled on a daily basis depends on the number of queries crawled and the number of posts these queries produce. If, for example, a certain tweet doesn’t match any of the (several hundred) issued daily queries, it is not picked up by the crawlers.

Besides the collection of posts and media data, we also check the language of each post because the different APIs don’t offer the same quality and reliability. Therefore, we use our own language recognition to verify the results. Furthermore, we validate that the crawled posts contain the correct combination of search terms according to the targeted query. The crawled posts/data are made available in a wide range of formats, such as JSON, XML, XLS, HTML.

4 FINANCIAL AND DEMOGRAPHIC DATA COLLECTION

In MULTISENSOR, we have tree different use case scenarios - journalism, commercial media monitoring and internationalisation, covered in D8.2. Each of them supports its own target group by providing different information and statistical data retrieved from the knowledge base. MULTISENSOR has a strategy for populating the semantic repository with data crawled from different sources, enriched by the Content Extraction Pipeline and converted to RDF. The problem with such data is that it is not enough to cover the needs of the main use cases. Therefore, we have performed an empirical study to find out what data would satisfy this purpose. As a result, we have chosen four datasets to serve as fundamentals of the knowledge base – DBpedia, Geonames, World Bank and Eurostat Indicators. In the next sections, we provide a brief overview of the datasets that are imported in the knowledge base, as well as an overview of the datasets that we plan to include during the next period.

4.1 Datasets populated in the knowledge base

4.1.1 DBpedia

The DBpedia dataset is created by extracting structured information from Wikipedia and presenting it in an RDF form (http://dbpedia.org/About). The conceptualization of the DBpedia dataset is based on the categories that are designed and implemented in Wikipedia, i.e. the data in the info-box section of the articles. This conceptualization is presented as ontology. For our purposes, we have used the English version 3.9. It contains:

- 4.58M things
- 1,445,000 persons
- 735,000 places
    - 478,000 populated places
- 411,000 creative works
- 241,000 organisations
- 251,000 species
- 6,000 diseases

This dataset will provide the base knowledge in our semantic repository. In future, we will also try to link the recognised entities from the Content Extraction Pipeline to the DBpedia concepts. This will provide fundamentals for our Decision Support system.

4.1.2 Geonames

Geonames is one of the central and most important geographical datasets in the Linked Open Data Cloud. It contains:

- 10M geographical names
- 9M unique features
    - 2.8M populated places
    - 5.5M alternate names

The stored data in this dataset includes latitude, longitude, population, administrative subdivision and postal codes. All coordinates use the World Geodetic System 1984

(WGS84). This dataset is important for MULTISENSOR because it will provide the needed geographical information.

4.2 Additional datasets to be populated in the knowledge base

4.2.1 World Bank

For MULTISENSOR purposes, we will use a subset of the World Bank dataset. It contains data from World Bank Indicators, World Bank Finances, World Bank Projects and Operations, and World Bank Climate Change, collected through the World Bank API endpoints. The World Bank dataset contains:

- 78M World Bank Climate Change triples
- 8M World Bank Finance triples
- 1M World Bank Projects and Operations
- 87M World Bank Indicators

These datasets will play a major role in the Decision Support system. Other sources of economic and commercial indicators that can be crawled are the following:

- https://www.cia.gov/library/publications/the-world-factbook/geos/gm.html
- http://en.wikipedia.org/wiki/Human_Development_Index
- http://data.oecd.org/leadind/business-confidence-index-bci.htm
- http://www.economywatch.com
- http://country-facts.findthedata.com/l/29/Germany
- www.tradingeconomics.com/germany/business-confidence
- www.indexmundi.com
- www.undata.com
- www.economy.com
- http://www.oecd.org
- http://www.imf.org
- https://www.destatis.de
- http://comtrade.un.org/
- http://www.bmwi.de
- http://www.focus-economics.com/countries/
- http://viewswire.eiu.com

4.2.2 Eurostat

The Eurostat data covers a number of areas from economy, through demographics, to trade and transport data. One can use it to learn about national statistics, explore industrial areas, and compare agricultural data across regions. In MULTISENSOR, we will choose only the indicators that will serve our needs.

Additional demographic and political indicators (e.g., government and the territorial distribution) can be retrieved from the sources bellow:

- https://www.cia.gov/library/publications/the-world-factbook/geos/gm.html
- http://data.worlkbank.org
- http://icex.es
- http://www.germany.info

- www.undata.com
- http://country.eiu.com/
- https://www.markit.com/Commentary/Get/13022015-Economics-German-economic-growth-smashes-expectations
- www.destatis.de

4.2.3 **Market Trends**

News, regulations, and market information are relevant to complement the pure statistical data and bring an added value that covers current issues, market trends, companies information, sector opportunities. Such information can be typically found in sources such as those bellow:

- http://www.ixpos.de
- http://www.bmel.de
- http://www.euromonitor.com/dairy-in-germany/report
- http://www.gtai.de
- http://www.ifs-certification.com
- http://www.hoovers.com/industry-facts.dairy-products-manufacturing.1354.html
- http://europa.eu/legislation_summaries/consumers/product_labelling_and_packaging/l21090_en.htm
- http://madb.europa.eu/madb/indexPubli.htm
- http://epp.eurostat.ec.europa.eu/portal/page/portal/statistics/search_database
- http://www.agenciatributaria.es/AEAT.internet/Inicio_es_ES/Aduanas_e_Impuestos_Especiales/Aduanas_e_Impuestos_Especiales.shtml

5 CRAWLER STORAGE INFRUSTRUCTURE

5.1 Storage for the crawled documents (CNR)

The Central News Repository (see D7.4, section 2.3.2.1) is the raw storage dump for the Crawlers (Site and Media collectors). The CNR is an Elastic Search[8] instance. This instance is used as a document pool to deliver the original documents in the Content Analysis Pipeline to be processed. Then, the CNR items are analysed by the CEP and the extracted/produced knowledge is stored in the RDF repository.

For each crawler a JSON API is provided with the structure of the document to be stored in the CNR (ElasticSearch). The news items (news articles, social media posts, etc.) are collected from the crawlers by calling the corresponding JSON API and become available in the CNR along with metadata (source, date, country, etc.). Then, the unprocessed news can be pulled by the analytic pipelines for processing. Table 4 shows the collected fields used for each crawler:

Field	Description	Used by the Site collector	Used by the Media collector
use_case	Name of the use case that the document refers to	X	X
Language	Language of the article	X	X
Country	Country from where comes from the article	X	X
_id	Identifier of the article in CNR	X	
Crawled	Date when the article was crawled.	X	X
multimediaUrls	List of the multimedia elements (url) available in the article	X	X
c_sourcecode	Content of the article in HTML format	X	X
Source	Media from where comes the articles	X	X
pr_summary	Summary provided with the article	X	
date_timestamp	Date of the article publication in the timestamp format	X	X
Body	Text of the articles	X	X

[8] https://www.elastic.co/

Feed	Name of the feed used to collect the article	X	
date	Date of the article publication	X	
article_id	Identifier of the article generated by the PR API	X	
pr_feed	Type of the feed used to collect the article (internet, press, files)	X	
title	Title of the articles	X	X

Table 4: List of the collected fields from the crawlers

5.2 Storage for multimedia metadata (CMR)

The Central Media Repository (CMR) is the storage of the source multimedia content (video, images and audio) collected by the harvester (see D7.4, section 2.3.2.2).

The CMR is built as a simple filesystem (Figure 4). Within the logic of the Crawler, a method is defined to operate asynchronous iterations over the "*multimediaUrls*" field. Inside this field there is a list of URLs that contain the related images, videos and audio files for a specific article retrieved by the Crawler. Because of this, customized folders are created for every article and save the multimedia content in that directory. The final goal of the CMR is to offer all that multimedia data to other services that could make used of it.

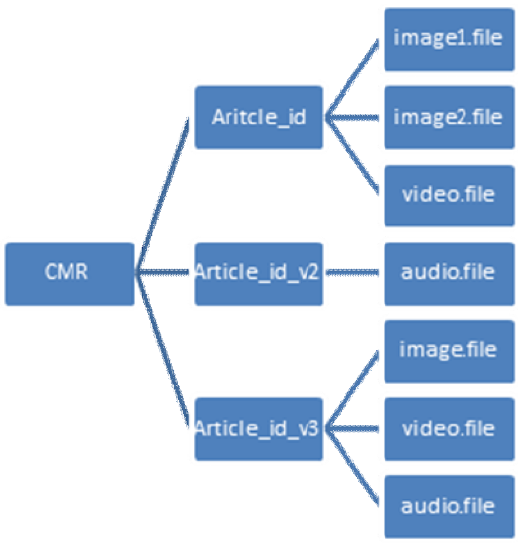

Figure 4: Folder structure to store the multimedia content in the CMR

At the end, the multimedia items are pulled and processed by the analytic pipelines for processing. Audio files are extracted from the video and text is extracted from the audio file.

6 CONCLUSIONS

In this deliverable we present the crawling infrastructure of MULTISENSOR and describe the implementations efforts that were conducted in the area of web crawling and social media data collection. We have presented our crawling architecture that consists of three main components. A collector for supporting real-time aggregation of sociometric counts, a crawler for crawling web pages using a seed list and API wrappers for retrieving structured data from knowledge bases.

The web crawler is based on the large scale Nutch open source crawler and is used both for focused crawling of web pages as well as for crawling multimedia objects. The store is implemented on top of Hadoop and HBase.

The deliverable provides information on the open source crawlers used, it describes the access to APIs of web crawlers for media article collection and finally it provides access to the code of the sociomertric collector.

For the next steps of the project we plan to integrate all components together and to have a full fledged store that contains all kind of data (status updates, web pages, multimedia and social network information). This store will serve as the entry point of all MULTISENSOR analytics pipeline. We will also include additional financial data as described in section 4. The final updates of the crawling infrastructure will be including in the upcoming prototype deliverables D7.6 and D7.7.

7 REFERENCES

Ahmed, M., Spagna, S., Huici, F., and Niccolini, S. (2013). *A peek into the future: Predicting the evolution of popularity in user generated content*. In Proc. 6th ACM Int'l Conf. Web Search and Data Mining, pages 607–616.

Brody, T., Harnad, S., and Carr, L. (2006). *Earlier web usage statistics as predictors of later citation impact: Research articles*. J. Am. Soc. Inf. Sci. Technol., 57:1060– 1072.

Freyne, J., Berkovsky, S., Daly, E. M., and Geyer, W. (2010). *Social networking feeds: recommending items of interest*. In Proc. 4th ACM Conf. Recommender Systems, pages 277–280.

Garimella, V. R. K. and Castillo, C. (2014). *Fast: Forecast and analytics of social media and traffic.* In Proceedings of the Companion Publication of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing, CSCW Companion '14, pages 13–16, New York, NY, USA. ACM.

Gupta, M., Gao, J., Zhai, C., and Han, J. (2012). *Predicting future popularity trend of events in microblogging platforms*. Proc. American Society for Information Science and Technology, 49(1):1–10.

Jamali, S. and Rangwala, H. (2009). *Digging digg: Comment mining, popularity prediction, and social network analysis*. In Proc. 2009 Int'l Conf. Web Information Systems and Mining, pages 32–38.

Kim, S.-D., Kim, S.-H., and Cho, H.-G. (2011). *Predicting the virtual tempera- ture of web-blog articles as a measurement tool for online popularity*. In Proc. 2011 IEEE 11th Int'l Conf. Computer and Information Technology, pages 449–454.

Lehmann, J., Gonçalves, B., Ramasco, J. J., and Cattuto, C. (2012). *Dynamical classes of collective attention in twitter*. In Proc. 21st Int'l Conf. World Wide Web, pages 251–260.

Lerman, K. and Hogg, T. (2010). *Using a model of social dynamics to predict popularity of news*. In Proc. 19th Int'l Conf. World Wide Web, pages 621–630.

Marujo, L., Bugalho, M., da Silva Neto, J. P., Gershman, A., and Carbonell, J. (2011). *Hourly traffic prediction of news stories*. In 3rd Int'l Workshop on Context-Aware Recommender Systems.

Mathioudakis, M., Koudas, N., and Marbach, P. (2010). *Early online identification of attention gathering items in social media*. In Proc. 3rd ACM Int'l Conf. Web Search and Data Mining, pages 301–310.

Pinto, H., Almeida, J. M., and Gonçalves, M. A. (2013). *Using early view patterns to predict the popularity of youtube videos*. In Proc. 6th ACM Int'l Conf. Web Search and Data Mining, pages 365–374.

Ruan, Y., Purohit, H., Fuhry, D., Parthasarthy, S., and Sheth, A. P. (2012). *Prediction of topic volume on twitter*. In Proc. 4th Int'l ACM Conf. Web Science.

Shamma, D. A., Yew, J., Kennedy, L., and Churchill, E. F. (2011). *Viral actions: Predicting video view counts using synchronous sharing behaviors*. In Proc. 5th Int'l Conf. Weblogs and Social Media.

Szabo, G. and Huberman, B. A. (2010). *Predicting the popularity of online content*. Communications of the ACM, 53:80–88.

Tatar, A., Leguay, J., Antoniadis, P., Limbourg, A., de Amorim, M. D., and Fdida, S. (2011). *Predicting the popularity of online articles based on user comments*. In Proc. Int'l Conf. Web Intelligence, Mining and Semantics, pages 67:1–67:8.

Tsagkias, M., Weerkamp, W., and de Rijke, M. (2010). *News comments: ex- ploring, modeling, and online prediction*. In Proc. 32nd European Conf. Advances in Information Retrieval, pages 191–203.

Tumasjan, A., Sprenger, T. O., Sandner, P. G., and Welpe, I. M. (2010). *Predicting elections with Twitter: What 140 characters reveal about political sentiment*. In Proc. 4th Int'l Conf. Weblogs and Social Media, pages 178–185.

Yu, B., Chen, M., and Kwok, L. (2011). *Toward predicting popularity of social marketing messages*. In Proc. 4th Int'l Conf. Social Computing, Behavioral-Cultural Modeling and Prediction, pages 317–324.

Zhang, X., Fuehres, H., and Gloor, P. A. (2011). *Predicting stock market indicators through Twitter "I hope it is not as bad as I fear"*. Procedia - Social and Behavioral Sciences, 26:55–62.