Computer Science • 14 (1) 2013 http://dx.doi.org/10.7494/csci.2013.14.1.45

Przemysław Maciołek
Grzegorz Dobrowolski

CLUO: WEB-SCALE TEXT MINING SYSTEM FOR OPEN SOURCE INTELLIGENCE PURPOSES

Abstract *The amount of textual information published on the Internet is considered to be in billions of web pages, blog posts, comments, social media updates and others. Analyzing such quantities of data requires high level of distribution – both data and computing. This is especially true in case of complex algorithms, often used in text mining tasks.*

*The paper presents a prototype implementation of CLUO – an Open Source Intelligence (*OSINT*) system, which extracts and analyzes significant quantities of openly available information.*

Keywords Text Mining, Big Data, OSINT, Natural Language Processing, monitoring

1. Introduction

Open Source Intelligence (*OSINT*) aims at presenting valuable information based on publicly available data. As it might be expected, the Internet is a primary (if not perfect) example of such data source. By applying text mining tools on a myriad of available services: online news, blogs, mailing lists, forums, portals, ... a great amount of insight might be provided into almost any topic.

While originally associated with governments, *OSINT* is also an area of interest for companies (making research on the market and/or the competition) or even personals (analyzing some specific topic). Some of the typical use cases include:

- collecting information on given topic (e.g. related to suspicious financial operations),
- searching information in given context (e.g. what were the financial operations of XYZ in 2009?),
- analyzing social networks, finding out connections between entities (e.g. *A* knows *B* and *B* knows *C*, which works for *XYZ Inc.*),
- analyzing gathered information (e.g. trends on how many articles about *XYZ* were published in 2009 and how many in 2008? [12]).

OSINT creates a *hard* problem in a computer supporting aspect, and requires much more implementation effort than, for example, keyword search engines.

- *OSINT* is a very broad task, covering multitude of aspects and workflows [2], so it is not easy (if not impossible) to provide a common tool for all *OSINT* use cases,
- *OSINT* tools are complicated itself, they cover both retrieving the information as well as searching, filtering, extracting and analysis,
- amount of processing done by *OSINT* system is typically much larger than that done by the typical search engine; very often, a large amount of semantic processing is included.

Currently, a number of dedicated *OSINT* tools is available, typically offered as an internet service rather than standalone application. Some of them are used only by government agencies, e.g. *MiTAP* (bio–security intelligence system [6]). Most of them are available commercially. The list of available systems is quite long and include: *Palantir*[1], *Recorded Future*[2], *Maltego*[3], *SiloBreaker*[4], and others.

Very often, only specific scope of analysis is supported by these applications (such as retrieving information from news services, analyzing brand sentiment basing on feedback on Twitter or other blogging services, etc.). *OSINT* practitioners use keyword search engines for finding intelligence information [1]. This is time consuming and requires manual analysis of retrieved data by an expert. But, on the other hand,

[1]`http://www.palantirtech.com/`
[2]`https://www.recordedfuture.com/`
[3]`http://www.paterva.com/web5/`
[4]`http://www.silobreaker.com`

many search engines provide access to very broad scope of data, with billions of web pages indexed.

This paper presents a prototype of *CLUO*, a custom built *OSINT* platform, with support of *PPBW (Polska Platforma Bezpieczeństwa Wewnetrznego)*. The motivation behind creating the system was to provide a tool that would cover large amount of data sources (e.g. not be limited only to few selected news sites), be easy to use and have a good support for European languages (starting with Polish and English). The software is used as a research platform for *PPBW* purposes and is being further developed by *Luminis Research*, with aim of providing commercial *SaaS (Software as a Service)* system.

2. Architecture

U.S. Department of Defense defines *OSINT* as *produced from publicly available information that is collected, exploited, and disseminated in a timely manner to an appropriate audience for the purpose of addressing a specific intelligence requirement* [3]. Such definition hints a number of design requirements:

- *OSINT* is actually a process, in which data must be first *collected* and then be a subject to *filtering* and *extraction*,
- system aims at being interactive, allowing users to retrieve required data in a *timely manner*,
- a significant number of use-cases (*specific intelligence requirements*) is supported.

The *OSINT* system can be broken down into three separate aspects:

1. **Collecting** – i.e. where to get the data from?
2. **Extracting and analyzing** – i.e. what the data contain?
3. **Presenting** – i.e. what does it mean?

Taking this into the realm of Internet–based data sources, a high-level architecture layout is proposed in fig. 1. The core of the presented system is split into three main functional parts – **Collecting Data**, **Preliminary Processing Chain** and **Request Processing Chain**. Such design is chosen because after receiving the data from the crawlers, computationally intensive tasks are started for extracting data [5, 11, 15] required by the last phase, which can be seen as a stack of filters. It works for the user's sake, is controlled by him and allows for retrieval of both the documents and the knowledge derived.

2.1. Collecting Data

Instead of relying on a single crawling solution, the system provides a *REST (REpresentational State Transfer)* style [10] interface, which allows for simple communication with any kind of data collection subsystem. A number of such processes asynchronously sends a list of retrieved documents. Each of the entries contains text, unique id and optional metadata fields, such as: *author name*, *publication date*, *language* or

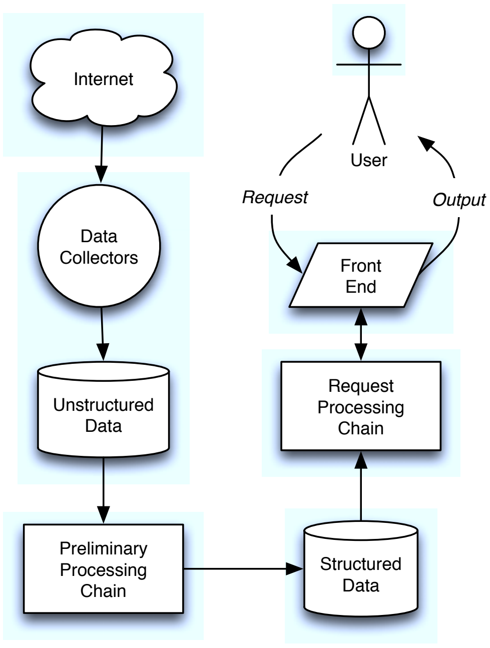


Figure 1. CLUO – High level architecture diagram.

thread id. In case they are not provided, the metadata values might be guessed later, in *Preliminary Processing Chain*.

In comparison to a solution tightly integrated with the system, using *REST* adds some communication overhead. However, it makes adding more data acquisition methods much simpler. They might be developed in any tool of choice, best suited for a particular task. The currently available list of data collectors consists of:

- **web crawler** – developed by *PPBW*, basing on *Apache Droids*[5]; the solution also supports a *meta-search* – in such case, a query is sent to a web-search engine and the results are used as a seed for the crawler,
- **forum data collector** – supports *phpBB*, *IPBoard* and *vBulletin*,
- **blog data collector** – uses *DOM (Document Object Model)* for extracting meaningful contents from engines such as *Blogspot*, *Wordpress*, *Blox.pl* and more,
- **social networks data collector** – supports *Facebook* and *Twitter*, via their API,
- **database collector** – allows for easily retrieving data from relational databases, as well as from various document collections; the latter is often used for testing and research based on widely available text corpora.

[5]`http://incubator.apache.org/droids/`

There are two modes the collectors might be run. The first one is *ad-hoc* operation. In such case, a collector might be run manually on a selected target, with provided constraints and parameters. The other one is based on a simple *message queue*, which provides list of starting addresses (a seed) with parameters. The data collectors periodically query the server and, if necessary, the acquisition is being started.

2.2. Preliminary Processing Chain

After retrieving a document, it is a subject of the preliminary processing chain. Its purpose is to retrieve the meaningful part of input contents and pre-process it, so that information will be extracted from the document and, in effect, other operations can be processed in a *timely manner*. The current implementation includes following stages, executed one after another:

1. **Content extraction** – the meaningful part of the content is extracted (i.e. elements such as navigational information or ads are removed).
2. **Language detection** – document language is automatically detected basing on used terms; this allows for use of a correct further processing method, optimized for the given language.
3. **Publication date extraction** – using metadata, custom developed heuristics and a dedicated classifier, the system is extracting the actual publication date.
4. **Text segmentation** – the text is split into sections, sentences and single terms.
5. **PoS tagging** – optionally, *part-of-speech* tagging is performed (it might be used as a hint by further processing).
6. **Stemming** – information about stem is tagged for each term [18].
7. **Sentiment tagging** – terms related to sentiment information are tagged; global sentiment information for given document is also calculated.
8. **Model building** – an internal document model is built. It includes not only the plain text contents, but also information about formatting, links, term metadata, etc.
9. **Named entity extraction** – named entities (such as person names, places, organizations, etc.) are automatically found and extracted.
10. **Indexing** – document is indexed (including document metadata, such as retrieval date, title, etc.).
11. **Classification** – tags are automatically assigned basing on the document contents.

2.3. Request Processing Chain

As soon as the application receives the user's request, e.g. to find specific documents or to build sentiment analysis reports, the **request processing chain** is executed. In general, it consists of three stages.

1. **Querying** – basing on constraints specified by the user, the documents are selected.
2. **Collecting** – depending on the request, appropriate information is collected (e.g. document contents, phrase occurrence count or sentiment information).
3. **Post-processing** – again, depending on the request type, the collected data are subject of post-processing transformations (e.g. in the case of sentiment analysis histograms are built in this step).

3. Implementation

Considering that system is developed for storing billions of documents, providing scalability was the essential requirement of all design decisions. To fulfill this requirement, the application is using *MapReduce* [7] distributed processing paradigm of sending *application to data* rather than *data to application*. This approach was popularized by the success of *Google*, whose indexing algorithms heavily relied on this architecture.

Several systems with similar approach have been developed, some of them freely available, such as: *Bixo*[6] or *Apache Nutch*[7] (remarkably, *Hadoop* – popular open-source *MapReduce* solution, also used in *CLUO* – was developed as a part of *Apache Nutch* and spun as a separate project later). While successful projects, both *Bixo* and *Apache Nutch* have a number of drawbacks when used for *OSINT* task:

- *Apache Nutch* is mainly a web search engine and crawler. While it might be extended for other tasks, when *CLUO* development started, it had limited options for supporting data sources other than web pages.
- *Bixo* provides crawling capabilities and a general batch workflow for processing data. However, it was concluded it is not suited for continuos operation (i.e. in which new documents might be added randomly, processed and stored permanently).
- Neither of them provides capabilities such as: document classification, sentiment analysis or named entity retrieval.
- The aim of the developed solution was to provide high quality information, with removal of unrelated elements (e.g. ads) and with information about author and publication date available for each document.
- The system needs quick access to much more than just documents and index. This include *named entities*, *cooccurrences*, *classifier models* or *term counts*. Considering the amount of data, this requires distributed storage.

While both *Apache Nutch* or *Bixo* could be used as a base for *OSINT* system development, they would have to be significantly changed and extended. It was concluded that building custom system from scratch, yet with significant reuse of open source components (where available), will allow to create the desired solution faster.

[6]`http://openbixo.org/`
[7]`http://nutch.apache.org/`

It will be also easier to implement custom data models and processing chain, better suited for the task. Following, several design decisions taken for *CLUO* are presented and discussed.

3.1. Lanugage-dependent processing

One of the first steps in **preliminary processing chain** is to automatically detect document language. This allows to use the right tools, optimized for given locale. Even if some of the steps are performed in exactly the same way for each language, some are vastly different. This specifically includes: *stemming*, *PoS tagging*, *named entity recognition*, *sentiment tagging* (both different heuristics and lexicon are used) and *indexing* (correct analyzer shall be selected).

In case of English, Stanford CoreNLP[8] suite is used [21, 20] for *PoS tagging* and *named entity recognition*. In contrary, the available options for similar tools for Polish had either unresolvable commercial licensing issues or were difficult to integrate and maintain in distributed *Hadoop* environment. Therefore, custom, Java-based solutions were developed, leveraging recent *NKJP* corpus [19]. The versions currently used in *CLUO* were trained using *Maximum Entropy* classifier from *Mallet* suite, basing on a widely used set of features: such as up to three previous words, their recognized parts-of-speech, etc.

A Polish *named entity recognition* results example is presented on Fig. 2. It shows how the trained tagger dealt with a fragment of regular news article, retrieved from *Gazeta.pl*[9]. As can be seen, it correctly recognized all mentioned persons, yet had some trouble with one of the organizations, which was split into two entities. It is worth noting that it also detects URL and e-mail addresses (using pattern matching) and contains heuristics for joining multi-word entities. The tagger is a subject for further improvement – mainly by collecting larger set of examples.

Z zaproszenia prof. ***Ryszarda Kaczmarka (PERSON)****, historyka i szefa* ***Instytutu Badań Regionalnych Biblioteki (ORGANIZATION) Śląskiej (PLACE)****, skorzystali specjaliści od socjologii, prawa i demografii. Wszyscy mieli krytyczne uwagi do sposobu, w jaki* ***Główny Urząd Statystyczny (ORGANIZATION)*** *przeprowadził zeszłoroczny spis powszechny. Reprezentująca tę instytucję dyrektor* ***Grażyna Witkowska (PERSON)*** *najczęściej zasłaniała się przepisami – polskimi i* ***unijnymi (ORGANIZATION)****. I krytykowała krytykujących, że nie skorzystali z okazji do konsultowania tych przepisów, gdy był ku temu czas. – Środowisko naukowe powinno wcześniej się nad tym pochylić, a nie płakać dziś nad rozlanym mlekiem – mówiła* ***Witkowska (PERSON)****.*

Figure 2. A sample of NER results for Polish, basing on a text retrieved from *Gazeta.pl*.

[8]`http://nlp.stanford.edu/software/corenlp.shtml`

[9]`http://wiadomosci.gazeta.pl/wiadomosci/1,114883,11617131,Naukowcy_skrytykowali_GUS_za_spis.html`

The system stores a summary of all term occurrences in a separate table. There is a single space for all supported languages, as it might be expected that there will be a number of common discriminating terms across the documents in various languages (e.g. proper names, foreign words, etc.) During classification stage, additional features might be extracted with help of *WordNet* [9] and *plWordNet* [17, 16].

3.2. Document Storage

The data store is leveraging *NoSQL* approach, rather than ubiquitous SQL relational database management engine. The main reason for this is that such solution allows to store structured data in a manner similar to the regular databases, but it still might be a subject of *MapReduce* jobs. Also, SQL solutions do not scale as well (and, more important, cheaply) as relatively simple distributed *NoSQL* solutions.

After decision has been made that document processing is realized using *Hadoop*[10], it was an easy design choice to select *HBase*[11] for storing processed contents. Alternatives to *HBase* includes: using other distributed database (such as *Cassandra*[12]) or storing data directly on a *DFS* (Hadoop's Distributed Filesystem). However, the first alternative does not seem to offer real improvement over *HBase*, while providing additional configuration effort. On the other hand, direct storage didn't have benefits of structured management of data and provided very limited capabilities for querying the documents.

As *HBase* has very limited indexing capabilities (and no full-text search), a custom solution was developed. The application applies *Lucene*[13] to index text data and few other selected fields. In effect, a dual approach for storing document data is introduced: the contents is accessed via database, but indexed via *Lucene*.

Such dualism allows to use very effective *Lucene* indexing for searching the data and very scalable *MapReduce* paradigm for processing documents, at the same time.

To show how such approach works in practice, lets consider a very common usage scenario, in which the documents are supposed to be retrieved basing on some user-specified query. The constraints are first passed to *Lucene*, which returns a list of matching *id's* (primary keys). Then, the *id's* are an argument of another query, executed on *HBase*, which effectively returns the requested data. As primary keys are sorted, such operation is very effective.

Currently, only single *Lucene* index is used, but it might be relatively easily distributed on a *Hadoop* cluster, either using shards or by leveraging approach such as *Katta*[14]. As the indexing system is pluggable, even more scalable engine could be also used, such as *IndexTank*[15].

[10] `http://hadoop.apache.org/`
[11] `http://hbase.apache.org`
[12] `http://cassandra.apche.org`
[13] `http://lucene.apache.org/`
[14] `http://katta.sourceforge.net/`
[15] `https://github.com/linkedin/indextank-engine`

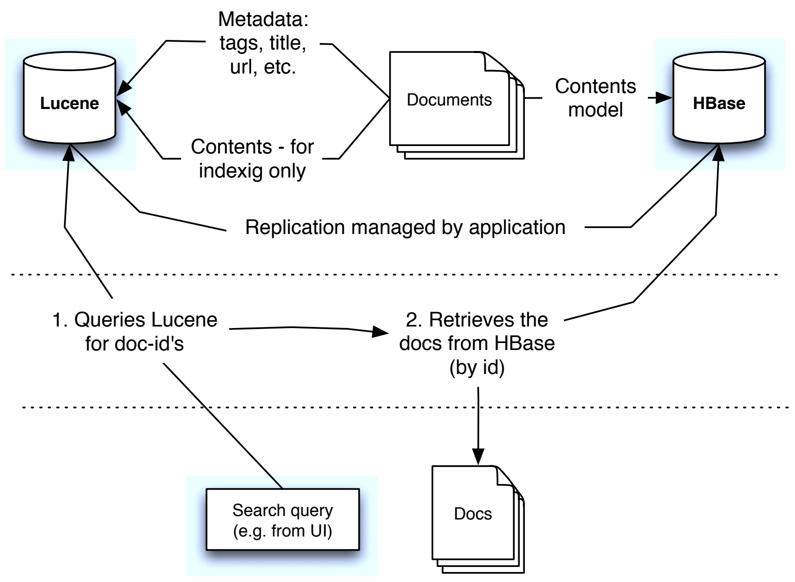


Figure 3. Storing documents in HBase and indexing them via Lucene.

3.3. Classifying

One of the crucial CLUO features is the classifying capability. It allows to automatically assign tags to new documents, basing on the history of previously known (*training*) documents and comparing their contents. A modular approach is chosen, which, depending on configuration, allows to use either traditional *bag-of-words model* or, more recently developed, *shallow semantic analysis graph model* [14]. In any case, the model features are extracted and internally stored as a *vector*, which is a subject of classification, performed with a help of *Mallet*[16] *MaxEnt* [4] classifier. Other engines were also considered, including *LIBSVM/LIBLINEAR*[17] and *Apache Mahout*[18]. However, *Mallet's MaxEnt* implementation significantly outperformed other methods during tests and the classifier was being built relatively quickly and did not require parameter tuning. Considering this and the fact that classifier building process could be easily distributed across the nodes basing on the target class, using *Apache Mahout* distributed classification capabilities was deemed unnecessary.

In *CLUO*, a binary tag classification model is used. For each single tag, a classifier is run, which tests the contents of the document and returns either *0* (not matching) or *1* (matching). The automatically classified documents can be manually verified by an user via a special *UI* widget. After that, they are included in the training set.

[16]`http://mallet.cs.umass.edu`
[17]`http://www.csie.ntu.edu.tw/~cjlin/`
[18]`http://mahout.apache.org/`

If the training set size increases over a certain threshold, the classifier is retrained, which should effect in classification quality improvement.

As an additional feature, *CLUO* internally also uses a special class, called *noise*. It contains documents such as *404 pages*, *redirections*, *advertisements*, etc. and is helpful in automatic removal of unwanted documents.

3.4. Hadoop Processing

One of the most important design decisions is to use *Hadoop* – an open source *MapReduce* [8] framework, initially designed with processing huge quantities of web documents in mind.

Although sometimes it is considered desirable to use some kind of *MapReduce* job abstraction layer – such as *Hive*[19] or *Pig*[20], it has been decided to use *Hadoop* directly. There are two main reasons for this: lack of good *HBase* support in these tools and the ability of having more control over job specification and the dependent tools.

This decision allowed also to develop a custom mechanism for assigning ranges of rows to the mappers split over cluster. In regular *HBase*, *MapReduce* operations are performed as batch jobs, which read all rows and filter them. While this fits well the typical nature of *Hadoop* processing, in *CLUO* often sparse data are being processed, i.e. just few thousand rows out of a billion-document database. As a remedy, custom *JobSplitter* was developed. Instead of filtering all rows, it reads matching IDs from the external index and assigns the processing to the right cluster members (the ones where data is actually present).

The performance of such solution depends mainly on the sparsity of retrieved data. Obviously, if huge part of all available data is being actually selected, then looking up in index slows down the whole process. Additionally, even if small part of data is to be retrieved, but spread out among all available underlying data blocks, then the benefit is minimal. This is due to the way data are stored by *HBase* using *HDFS (Hadoop Data File System)*. When retrieving a row, the whole data block (by default 100 MB) is being read from the disk. To solve this problem, good strategy for creating primary IDs should be used. In case of *CLUO*, the ID is based on the URL, as it might be expected that data regarding similar topics will be often grouped.

Currently, the preliminary processing chain consists of three *Hadoop* jobs (as described in Fig. 4) and one additional job, which indexes the processed contents. The choice of the keys and values for *Map* and *Reduce* inputs and outputs reflects the most effective processing split, in which minimal additional amount of data has to be transferred between processing nodes.

As for the request processing chain, depending on the scope and expected amount of data, some operation are executed on a single (main server) machine and the others are run as *MapReduce* jobs.

[19]`http://hive.apache.org`
[20]`http://pig.apache.org`

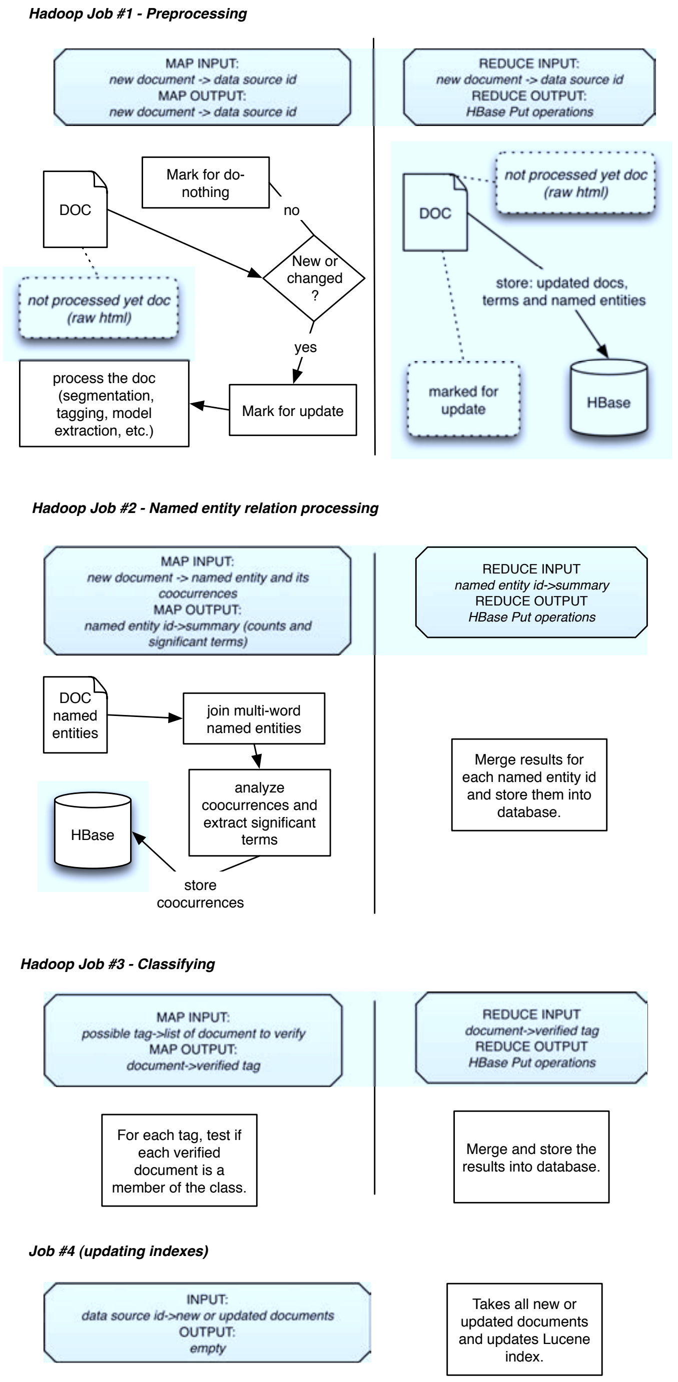

Hadoop Job #1 - Preprocessing
MAP INPUT:
new document -> data source id
MAP OUTPUT:
new document -> data source id
REDUCE INPUT:
new document -> data source id
REDUCE OUTPUT:
HBase Put operations
Mark for do-nothing
no
DOC
New or changed ?
not processed yet doc (raw html)
yes
process the doc (segmentation, tagging, model extraction, etc.)
Mark for update
DOC
not processed yet doc (raw html)
store: updated docs, terms and named entities
marked for update
HBase
Hadoop Job #2 - Named entity relation processing
MAP INPUT:
new document -> named entity and its coocurrences
MAP OUTPUT:
named entity id->summary (counts and significant terms)
REDUCE INPUT
named entity id->summary
REDUCE OUTPUT
HBase Put operations
DOC named entities
join multi-word named entities
analyze coocurrences and extract significant terms
HBase
store coocurrences
Merge results for each named entity id and store them into database.
Hadoop Job #3 - Classifying
MAP INPUT:
possible tag->list of document to verify
MAP OUTPUT:
document->verified tag
REDUCE INPUT
document->verified tag
REDUCE OUTPUT
HBase Put operations
For each tag, test if each verified document is a member of the class.
Merge and store the results into database.
Job #4 (updating indexes)
INPUT:
data source id->new or updated documents
OUTPUT:
empty
Takes all new or updated documents and updates Lucene index.

Figure 4. Jobs used in CLUO preliminary document processing chain.

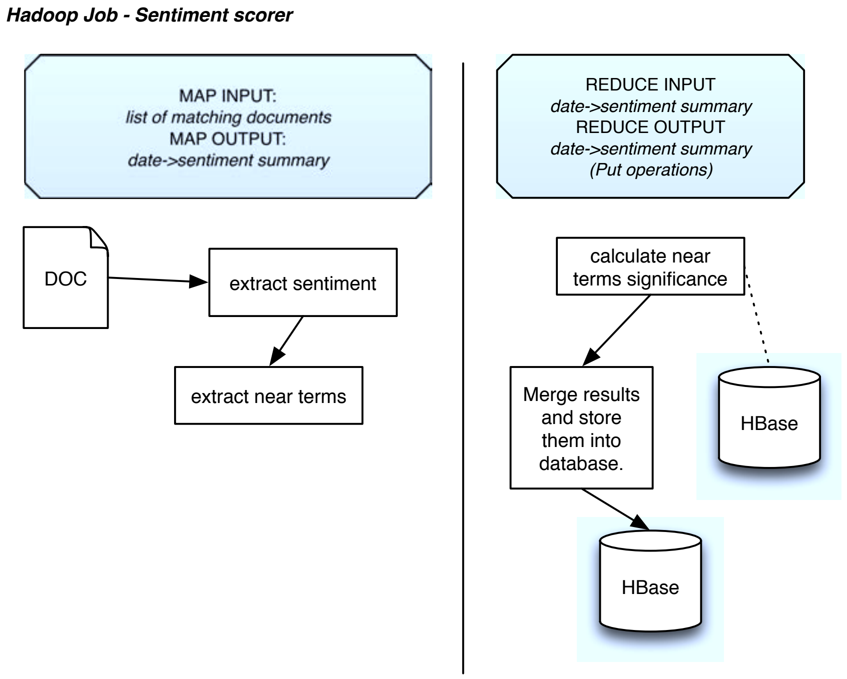


Figure 5. Sentiment scorer job.

In the first case, this includes:

- searching and browsing specific documents,
- browsing named entities and connections between them,
- tagging documents and verifying the classified documents.

Currently, there is only one job type in the request processing chain – *sentiment scorer job* (Fig. 5). It is processing the data using selected keywords, analyzes sentiment (either related to the keywords or general), finds related terms, common authors and trends – all in the same time. It is worth noting that for near terms extraction, the algorithm is in fact constructing association lists [13], with finding candidates in *mapper* and selecting them in *reducer*.

The sentiment analysis, itself, is a very broad conception, also known as polarity analysis or opinion mining. In general, its aim is to determine the attitude of a writer (or speaker) on some subject. In the simplest case, expressed as either "positive", "neutral" or "negative". There is a number of approaches that might be used for this task. The current version of *CLUO* is using statistical analysis, with a help of polarity dictionary and a number of hand-written rules. There is an ongoing work on collecting a corpus and building a machine learning-based solution.

4. Practical tests

The prototype version of the system was deployed on a small cluster, hosted externally, consisting of 3 quad-core Intel i5 2500K processors. The largest tested database has contained approx. one billion of documents, downloaded during a span of few days from selected Internet foras, blogs and news sites. Several observations were made:

- The bottleneck is actually downloading and crawling the data from Web and social media services. While this could be easily distributed for small websites, it is more complicated to effectively fetch a huge one, due to the fact that synchronization mechanism would have to be developed for parallel crawlers working on the same website.
- Preprocessing the data scales linearly and is being done in rate of between 10 (long documents, full *PoS* and *NE* tagging) and 10 000 (very short documents, limited or no tagging) of docs per second per single core.
- The current implementation of sentiment scorer job is processing the documents at a rate of approx. 100 documents per second per core. It also scales almost linearly.
- The cluster size should be be determined basing on the required processing power rather than disk space. One billion documents, including all artifacts and metadata took only approx. 200 GB of space.
- Using custom indexing typically improved the speed of jobs (when about 1000 – 100 000 of documents were retrieved) by a factor of 100 (due to the fact that only matching documents were loaded from the database rather than all with filtering occurring later). The improvement for search queries (when only 50 docs are retrieved) is even higher, as the data were retrieved almost immediately.

Following figures present sample screens, showing report on London 2012 Olympic Games. Out of the gathered documents, approx. 100 000 were related to the recent games. For each post, the sentiment was analyzed and related terms (association lists) were extracted. Figure 6 presents volume (using bars) and percentage amount of negative (red line) and positive (green line) mentions. As can be clearly seen, positive ones dominate most of the time. However, on one occasion (Aug 7th), the balance changed. After clicking on the bar, system presents list of top authors (Fig. 7) and tag cloud with hot terms (Fig. 8). On that particular day, it is very significantly dominated with *Saeid Abdvali* mentions (additionally, the cloud is mostly in dark gray color, which denotes negative sentiment). Clicking on any of the tag cloud entries shows sample contexts (Fig. 9), which reveals that a lot of users were criticizing referee decision regarding *Saeid Abdvali* defeat in a wrestling match.

To retrieve such results, an user is required only to provide the key terms related to the subject of research. In the presented sample, this was: `‘‘olympics, london 2012’’` after `Jul 15th`, without additional constraints. Running this simple query quickly reveals interesting facts and anomalies. *CLUO* might also return results without specifying any keywords. It can perform analysis using any kind of constraint, be it a data source, a category, an author, etc.

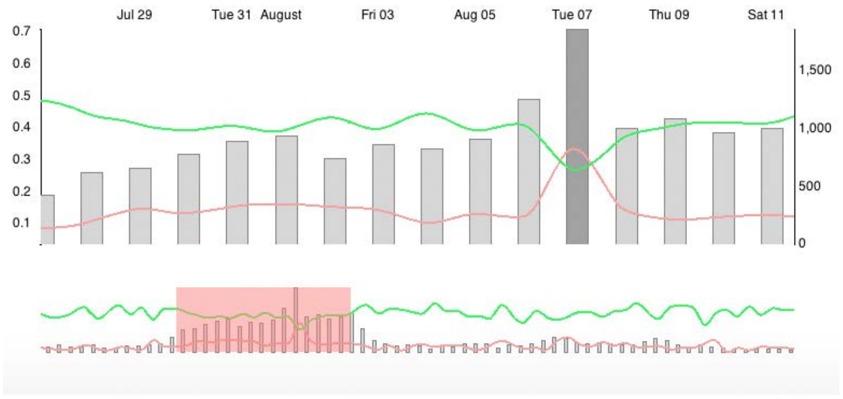


Figure 6. London 2012 Olympics report – average sentiment over time.

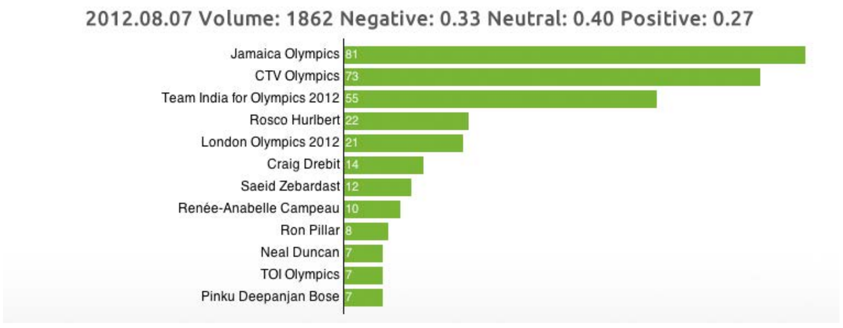


Figure 7. London 2012 Olympics report – top authors on Aug 7th.

Table 1 shows a small excerpt of *named entity* cooccurrence results. In this example, *named entities* related to *Beyonce* (American singer-songwriter) are displayed. The table shows organizations, places, internet addresses or names that were found in the same context, as well as the key terms and extracts from the text. This allows to find, among other relations, that *ABC* is related via an *interview* and *House of Dereon* via *tickets* and *NYC*. Such list is currently presented using a “raw” table, but in the future versions of *CLUO* it is going to be visualized via a graphical network.

5. Summary

In this paper a practical approach to building a text mining solution for *Open Source Intelligence* purposes has been described. *OSINT* is characterized by the need of

Table 1
A sample presenting named entity cooccurrences.

Name	**Type**	**Top terms**	**Context**
House of Deron	`ORGANIZATION`	giving, NYC, pair, concert, Sunday, film, 11, 20, tickets, away	...House of Deréon is giving away a pair of tickets to the exclusive screening with Beyonc of her Live at Roseland concert film in NYC Sunday, 11 20! ...
Jumpin Jumpin Independent Women Part	`ORGANIZATION`		...
Katie Couric	`PERSON`	deluxe, everything, disc, Elements, 20, sits, interview, speak, her, release	...Tonight Beyoncé sits down with Katie Couric on 20 20 for an intimate interview to speak about everything from her new Live At Roseland: Elements of 4 deluxe 2 disc DVD release to her pregnancy. ...
ABC	`ORGANIZATION`	deluxe, 10pm, everything, disc, Elements, ET, 20, sits, interview, speak	...Tonight Beyoncé sits down with Katie Couric on 20 20 for an intimate interview to speak about everything from her new Live At Roseland: Elements of 4 deluxe 2 disc DVD release to her pregnancy. Make sure you tune in to ABC tonight at 10pm ET to watch. ...
`www.smarturl.it/liveatroseland`	`INTERNET ADDRESS`	Live, Elements, deluxe, Pre, available, DVD, everywhere, holiday, gift, perfect	...Watch Beyoncé rock the stage with Love On Top from her deluxe 2 – DVD set Live At Roseland: Elements of 4 available everywhere now: http://www.smarturl.it/liveatroseland ...

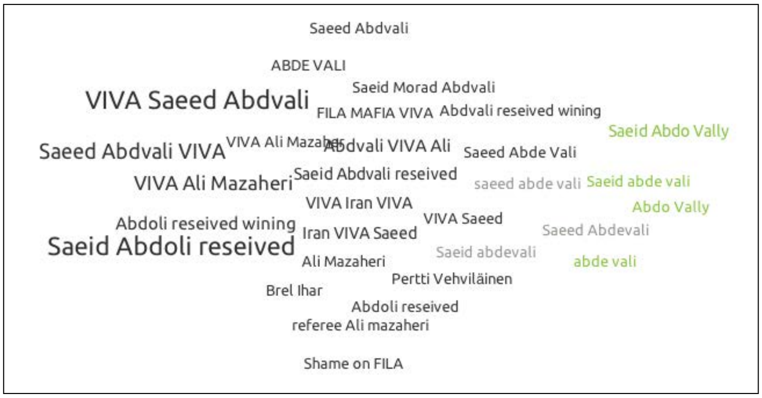
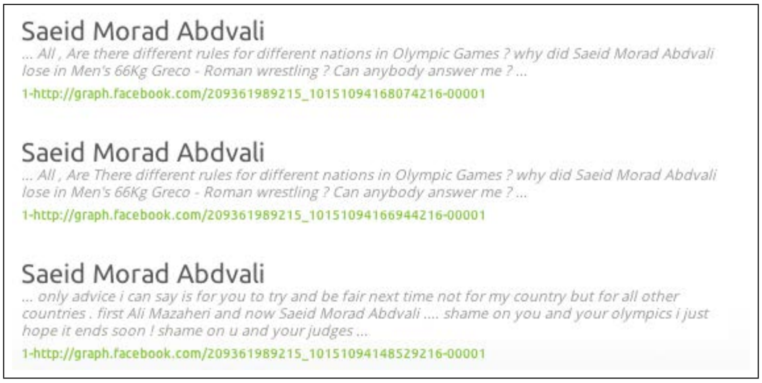


Figure 8. London 2012 Olympics report – hot terms on Aug 7th.


Saeid Morad Abdvali
... All , Are there different rules for different nations in Olympic Games ? why did Saeid Morad Abdvali lose in Men's 66Kg Greco - Roman wrestling ? Can anybody answer me ? ...
1-http://graph.facebook.com/209361989215_10151094168074216-00001

Saeid Morad Abdvali
... All , Are There different rules for different nations in Olympic Games ? why did Saeid Morad Abdvali lose in Men's 66Kg Greco - Roman wrestling ? Can anybody answer me ? ...
1-http://graph.facebook.com/209361989215_10151094166944216-00001

Saeid Morad Abdvali
... only advice i can say is for you to try and be fair next time not for my country but for all other countries . first Ali Mazaheri and now Saeid Morad Abdvali .... shame on you and your olympics i just hope it ends soon ! shame on u and your judges ...
1-http://graph.facebook.com/209361989215_10151094148529216-00001

Figure 9. London 2012 Olympics report – context samples for *Saeid Abdvali* on Aug 7th.

processing huge amount of data, almost in real time, to provide meaningful insight into the area of user interest.

To handle this, presented prototype integrates a number of natural language processing tools and methods, implementing them in a *MapReduce* environment. This yields good results in providing business value, by effectively processing huge datasets in a distributed manner.

Hadoop – a *MapReduce* platform – has been chosen as the basis of the system, with addition of *Lucene* as an indexing tool and a distributed *NoSql* database – *HBase*. The paper presented a rationale for this choice, as well as the structure of designed jobs included in the processing chain.

End users are presented with easy to use interface, which allows them to quickly and simply run text mining jobs, providing valuable results, that can be interactively analyzed and browsed.

The system evaluation supports the conclusion the chosen design is effective and gives vast possibilities to scale with the amount of data processed.

In the future the addition of more text mining and data mining algorithms are planned, as well as optimization of current tasks and improving crawling performance and capabilities.

Acknowledgements

The research has received support and sponsorship from Luminis Research Sp. z o.o.

The research leading to these results has received funding from the European Communitys Seventh Framework Program (FP7/2007-2013) under grant agreement nr 218086.

References

[1] *NATO Open Source Intelligence Handbook.* NATO, 2001.

[2] *NATO Intelligence Exploitation of the Internet.* NATO, 2002.

[3] *National Defense Authorization Act for Fiscal Year 2006.* 2006.

[4] Berger A. L., Pietra V. J. D., Pietra S. A. D.: A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22(1):39–71, March 1996.

[5] Cover T., Thomas J.: *Elements of Information Theory.* Wiley, 1991.

[6] Damianos L. E., Ponte J. M., Wohlever S., Reeder F., Wilson D. G., Hirschman L.: Mitap, text and audio processing for bio-security: A case study. In *National Conference on Artificial Intelligence*, pp. 807–814, 2002.

[7] Dean J., Ghemawat S.: Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[8] Dean J., Ghemawat S.: Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51:107–113, January 2008.

[9] Fellbaum C.: *WordNet – An Electronic Lexical Database.* The MIT Press, 1998.

[10] Fielding R. T.: *Architectural styles and the design of network-based software architectures.* PhD thesis, 2000.

[11] Jurafsky D., Martin J. H.: *Speech and Language Processing* Prentice Hall, 2 ed., 2008.

[12] Leskovec J., Backstrom L., Kleinberg J.: Meme-tracking and the dynamics of the news cycle. In *Proc. of the 15th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, KDD '09, pp. 497–506, New York, NY, USA, 2009. ACM.

[13] Lubaszewski W., Gajęcki M.: Automatic extraction of semantic association from polish text. *Computer Science*, 4:119–130, 2002.

[14] Maciolek P., Dobrowolski G.: Is shallow semantic analysis really that shallow? a study on improving text classification performance. In *IMCSIT*, pp. 455–460, 2010.

[15] Manning C., Raghavan P., Schütze H.: *Introduction to Information Retrieval.* Cambridge University Press, 1 ed., 2008.

[16] Maziarz M., Piasecki M., Szpakowicz S.: Approaching plWordNet 2.0. In *Proc. of the 6th Global Wordnet Conference*, Matsue, Japan, January 2012.

[17] Piasecki M., Szpakowicz S., Broda B.: *A Wordnet from the Ground Up.* Oficyna Wydawnicza Politechniki Wroclawskiej, Wroclaw, 2009.

[18] Porter M. F.: An algorithm for suffix stripping. *Program*, 1980.

[19] Przepiórkowski A., Bańko M., Górski R. L., Lewandowska-Tomaszczyk B., eds. *Narodowy Korpus Języka Polskiego [Eng.: National Corpus of Polish].* Wydawnictwo Naukowe PWN, Warsaw, 2012.

[20] Toutanova K., Klein D., Manning C., Singer Y.: Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. of HLT-NAACL 2003*, 2003.

[21] Toutanova K., Manning C. D.: Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proc. of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 2000.

Affiliations

Przemysław Maciołek
Luminis Research Sp. z o.o., Rzeszów, Poland and AGH University of Science and Technology, Krakow, Poland, `pmaciolek@luminis-research.com`

Grzegorz Dobrowolski
AGH University of Science and Technology, Krakow, Poland, `grzela@agh.edu.pl`

Received: 17.08.2012
Revised: 24.09.2012
Accepted: 3.12.2012