A Distributed Framework for NLP-Based Keyword and Keyphrase Extraction From Web Pages and Documents

P. Nesi, G. Pantaleo and G. Sanesi
Distributed Systems and Internet Technology Lab, DISIT Lab, http://www.disit.dinfo.unifi.it
Department of Information Engineering (DINFO),
University of Florence – Firenze, Italy
paolo.nesi@unifi.it, gianni.pantaleo@unifi.it

***Abstract*—The recent growth of the World Wide Web at increasing rate and speed and the number of online available resources populating Internet represent a massive source of knowledge for various research and business interests. Such knowledge is, for the most part, embedded in the textual content of web pages and documents, which is largely represented as unstructured natural language formats. In order to automatically ingest and process such huge amounts of data, single-machine, non-distributed architectures are proving to be inefficient for tasks like Big Data mining and intensive text processing and analysis. Current Natural Language Processing (NLP) systems are growing in complexity, and computational power needs have been significantly increased, requiring solutions such as distributed frameworks and parallel computing programming paradigms. This paper presents a distributed framework for executing NLP related tasks in a parallel environment. This has been achieved by integrating the APIs of the widespread GATE open source NLP platform in a multi-node cluster, built upon the open source Apache Hadoop file system. The proposed framework has been evaluated against a real corpus of web pages and documents.**

Keywords – Natural Language Processing, Part-of-Speech Tagging, Parallel Computing, Distributed Systems.

I. INTRODUCTION

Nowadays we are living in a hyper-connected digital world, dealing with computational solutions progressively more oriented to data-driven approaches, due to the increasing population of web resources and availability of extremely vast amounts of data [1]. This fact has opened new attractive opportunities in several application areas such as e-commerce, business web services, Smart City platforms, e-Healthcare, scientific research, ICT technologies and many others. Modern information societies are characterized by handling vast data repositories (both public and private), according to which Petabyte datasets are rapidly becoming the norm. For instance, in 2014 Google has been estimated to process over 100 Petabytes per day on 3 million servers[1]; in the first half of 2014, Facebook warehouse has been reported to store about 300 PB of Hive data (with an incoming daily rate of about 600 TB)[2]. These numbers reveal how the process of storing data is rapidly overcoming our ability to process them in an automatic and efficient way. Such a huge pool of available information has attracted a lot of interest within several research and commercial scenarios, ranging from automatic comprehension of natural language text documents, supervised document classification [2], content extraction and summarization [3], design of expert systems, recommendation tools, Question-Answer systems, query expansion [4], up to social media mining and analysis, in order to collect and assess users' trends and habits for target-marketing, customized services. Therefore, applications and services must be able to scale up to items, domains and data subset of interest. Approaches based on parallel and distributed architectures are spreading also in search engines and indexing systems; for instance, the ElasticSearch[3] engine, which is an open source distributed search engine, designed to be scalable, near real-time capable and providing full-text search capabilities [5].

NLP systems are commonly executed in a pipeline where different plugins and tools are responsible for a specific task. One of the most intensive tasks is annotation, defined as the process of adding linguistic information to language data [6], usually related to their grammatical, morphological and syntactical role within the given context. Text annotation is at the basis of higher level tasks such as extraction of keywords and keyphrases (which deals with the identification of a set of single words or small phrases representing key segments that can describe the meaning of a document) [7], up to the design of Semantic Computing frameworks (involving activities such as supervised classification of entities, attributes and relations), that lead to the production of structured data forms, typically in the form of taxonomies, thesauri and ontologies. Current sequentially integrated NLP architectures, where each pipeline step uses intermediate results and outcomes produced by previous processing stages, are prone to problems related with information flow, from congestion to information losses [8]. It

[1] http://www.slideshare.net/kmstechnology/big-data-overview-2013-2014.

[2] https://code.facebook.com/posts/229861827208629/scaling-the-facebook-data-warehouse-to-300-pb/.

[3] https://www.elastic.co/products/elasticsearch

has been often addressed in literature how existing NLP tools and frameworks or are not well suited to process very large corpora, since their primary design focus was not oriented to scalability [9]. The only reasonable approach to handle Big Data problems seems to be the well-known computer science concept of "*Divide and Conquer*" [10]. The basic idea is to partition a large problem into smaller sub-problems; to the extent that these sub-problems are independent, they can be parallely processed by different threads. This aspect, concurrently with the evolution of distributed systems multi-processor computer architectures and network speeds, is leading to the application of distributed architectures and parallel-computing paradigms to Big Data mining and processing activities.

Keywords and keyphrases annotation can be useful for content extraction and summarization, in order to produce machine-readable corpora, as well as building content-based multi-faceted search queries. For instance, scientific articles are often annotated with keywords, in a way similar as it happens with metadata annotation of multimedia resources. Conversely, expanding our view to the available online resources, currently a large portion of web documents still does not have any keywords or keyphrases assigned. Besides, since manual annotations results to be an extremely time-consuming and inefficient process, the necessity arises to design efficient and scalable automated solutions to extract keywords and keyphrases from massive amounts of unstructured text documents.

The present work describes a novel framework which allows the execution of general NLP tasks, through the use of the open source GATE[4] tool [11], on a multi-node cluster based on the open source Apache Hadoop[5] Distributed File system (HDFS). The paper is organized as follows: Section II illustrates related work, in terms of state of the art and open issues for both commercial and research literatures; in Section III, an architectural overview of the proposed system is presented; in Section IV, a validation of the system, performed against a real corpus of web resources, is reported; finally, Section V is left for conclusions and future perspectives.

II. Related Work

The task of Automatic keyword extraction has been extensively studied in literature. Existing methods are typically divided into four categories: simply statistic, linguistic, machine learning and other mixed approaches [12]. Statistical methods usually relies on term position, term frequency, co-occurrence and related relevance metrics, such as TF-IDF [13]. The linguistic approach is based on NLP techniques, such as Part-of-Speech (POS) tagging, lexical analysis, syntactic analysis, possibly exploitation of semantic features [14]. Machine learning methods treat the keyword extraction task as supervised learning problem using a training dataset, using different techniques such as naive Bayes algorithms [15], least square support vector machines (LS-SVM) [16], etc. Other mixed approaches mainly use a combination of the previously described techniques, possibly adding some heuristic knowledge, for instance the use of annotated lists, gazetteers, blacklists to remove stop words [17], selecting only certain part-of-speech tags as candidate keywords [18]. Moreover, external resources (in addition to training corpora) can be used as lexical knowledge bases, such as Wikipedia [19] or DBpedia. Graph-based approaches typically extract a graph from each input documents and use a graph-based ranking function to determine the relevance of the nodes, which yields the importance of key terms [20]. Topic-based clustering is used in content extraction and text summarization methods; this usually involves grouping the candidate keyphrases into topics or domains [21], [22].

We find first attempts of employing parallel computing frameworks for NLP tasks in the middle 90s: Chung and Moldovan [23] proposed a parallel memory-based parser called PARALLEL, implemented on a parallel computer, the Semantic Network Array Processor (SNAP). Later, Van Lohuizen [24] proposed a parallel parsing method relying on a work stealing multi-thread strategy in a shared memory multi-processor environment. Hamon et al. realized Ogmios [9], a platform for annotation enrichment of specialized domain documents within a distributed corpus. The system provide NLP functionalities such as word and sentence segmentation, named entity tagging, POS-tagging and syntactic parsing. Jindal et al. [25] developed a parallel NLP system based on LBJ [26], a platform for developing natural language applications, and Charm++ [27] as a parallel programming paradigm. Exner and Hugues recently presented Koshik [28], a multi-language NLP processing framework for large scale-processing and querying of unstructured natural language documents distributed upon a Hadoop-based cluster. It supports several types of algorithms, such as text tokenization, dependency parsers, coreference solver. The advantage of using the Hadoop distributed architecture and its programming model (known as MapReduce), is the capability to efficiently and easily scale by adding inexpensive commodity hardware to the cluster.

We find also commercial tools aiming at solving the addressed problem. Beemoth[6], produced by Digital Pebble, is an open source platform for large scale document processing based on Apache Hadoop, employing third party NLP tools, including GATE, Tika and UIMA. InfoTech RADAR[7] is a software solution implementing NLP and Sentiment Analysis on the Hortonworks Sandbox Hadoop environment. Some existing NLP tools have proposed improvements to realize large-scale processing solutions; this is the case of the GATECloud project [29], which is an adaptation of the GATE software suite to a cloud computing environment (using the PaaS paradigm), although it is delivered under a fee payment, unlike the original platform.

Recently, other parallel computing frameworks have been proposed in addition to the Hadoop MapReduce. For instance, Spark framework [30] has been originally developed at Berkeley UC to support a wider class of applications,

[4] https://gate.ac.uk/

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

[6] https://github.com/DigitalPebble/behemoth/wiki/tutorial

[7] http://www.itcinfotech.com/software-product-engineering/solutions/RADAR.aspx

especially those implementing acyclic data flow models, such as iterative algorithms and applications requiring low-latency data sharing processes. Spark introduces programming transformations on Resilient Distributed Datasets (RDD) [31], which are read-only collections of objects distributed over a cluster of machines providing fault tolerance by rebuilding lost data by using lineage information (without requiring data replication). RDD allows to store data on memory, as well as to define the persistence strategy. Gopalani and Arora [32] have compared Spark and Hadoop performances on clustering K-means algorithms, showing that Spark outperforms Hadoop on different cluster configurations.

III. System Architecture

As introduced earlier, the proposed system aims at extracting keywords and keyphrases from web resources (retrieved by crawling online web pages and documents of business entities and research institutes) in a distributed architecture. The necessity of realizing a more efficient and scalable solution, in order to improve performances, as well as data integrity and failures handling, led us to the choice of the open source Apache Hadoop framework, which have been installed on a multi-node cluster. The Hadoop ecosystem is implemented in Java, and it is capable of supporting distributed

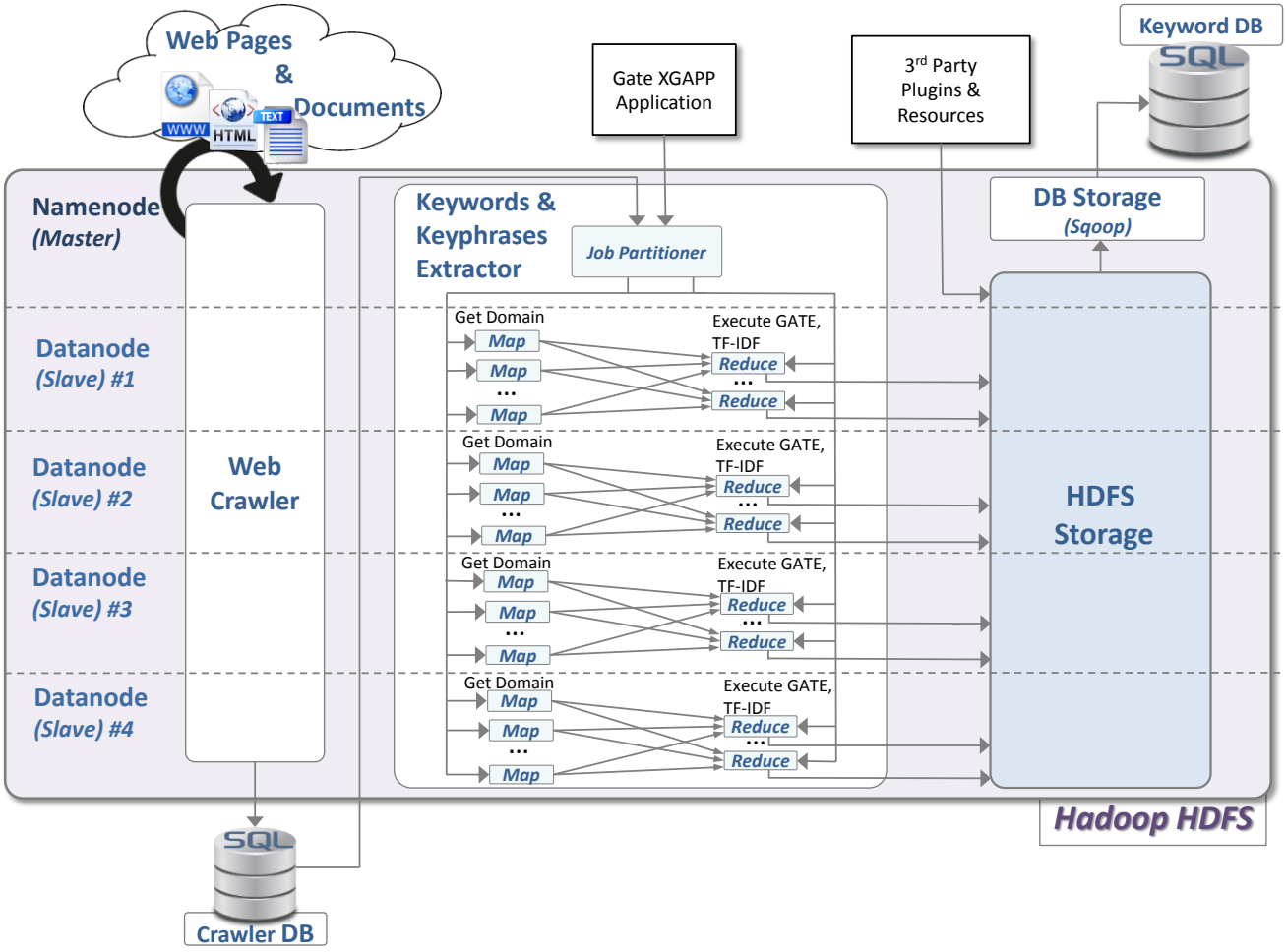

Figure 1. Overview of the proposed system architecture.

applications, large-scale data processing and storage, providing high scalability. Data access and management relies on the Hadoop Distributed File System (HDFS), modeled upon the Google File System – GFS (Ghemawat et al., 2003). Typically, a cluster is composed by a master Namenode, which assigns tasks and tracks their execution to the different clients (Datanodes). Datanodes are responsible also for data storage through its MapReduce programming model.

In our context, MapReduce is used to parallelize the crawler work, the execution of NLP tasks and final SQL database storage. In our case, NLP tasks are performed by using the GATE Embedded Java APIs, and they are related to the extraction of keywords and keyphrases from online parsed unstructured text, although our approach is general enough to potentially execute any generic GATE-based application. At the end of the distributed process (involving web crawling, text parsing, annotation, keywords/keyphrase extraction and pruning, based on their relevance), the produced outcome is stored in the HDFS file system. Finally, a dedicated procedure stores designed keywords and keyphrases in an external SQL

database. The whole system architecture is implemented in Java. In the next subsections, a description will follow of the main modules constituting the proposed framework, which are listed as following:

- The *Web Crawler* module.
- The *Keywords/Keyphrases MapReduce Extractor* module, which is responsible of two main operations:
  - The execution of the GATE application via MapReduce and the subsequent storage of extracted keywords/keyphrases in the HDFS.
  - The keywords and keyphrases relevance estimation, obtained by computing the TF-IDF function for each extracted keywords/keyphrases, performed to assess their relevance with respect to their whole corpus (for filtering purposes).
- The *DB Storage* module which finally stores designed keywords and keyphrases into an external SQL database.

An overview of the proposed architecture is depicted in Figure 1. The next subsection are committed to describe in further details the above listed modules.

A. Web Crawler

The crawling engine of the proposed system is based on the open source Apache Nutch[8] tool. It has been initialized with a set of seed URLs of commercial companies, services and research institutes. Actually, as later addressed as an open issue for future work, a future goal will be to exploit the capabilities of the proposed system in annotating text relevant features for content extraction and summarization (in order to estimate, as an example, the domain of interest of analyzed web domains, as well as the main key topics).

The Nutch based crawler workflow is divided into the following phases: The first is the *Inject* phase, in which some seed URLs are injected for initial bootstrapping. Then, in the *Generate* phase, a generating algorithm produces a set of URLs that are going to be fetched. The *Fetch* phase deals with fetching the previously generated set of URLs into segments. Subsequently, the *Parse* phase is dedicated to the parsing of fetched segments content; the *Update* phase populates an external SQL Database with parsed segment contents. Finally, the Apache Solr[9] technology is used to index all the collected documents, providing also a search interface. The Solr index is ultimately present only onto the master Namenode.

B. Keywords/Keyphrases MapReduce Extractor

This module reads and takes as input the URLs (stored in the SQL database). The content of the corresponding parsed web page is obtained through a query to the Solr index. In this way, documents are created for each corpus represented by a single web domain. The crawler and keywords extractor modules work asynchronously, actually the crawling task and the extraction process are scheduled and executed independently. Moreover, the present module is in charge of defining the MapReduce model. Typically, Map functions divide the work into smaller jobs or file blocks, which are subsequently mapped among the different Datanodes by the Namenode. Specific Map functions are defined to generate key/value pairs representing logical records from the input data source. Subsequently, Reduce tasks merge all intermediate values associated with the same key. The Hadoop services in charge of managing and assigning tasks and data blocks to the different nodes are the *JobTracker* and the *TaskTracker*. The former communicates with the Namenode to retrieve data locations and the directory tree of all files in the file system, in order to find available nodes and set specific tasks to assign. The latter represents a node in the cluster that is able to receive MapReduce tasks from a JobTracker; each Tasktracker is configured with a set of slots, indicating the number of tasks that can be accepted). Here, Map and Reduce functions are defined in a proper way for our goals: since we are interested in annotating keywords and keyphrases at single web-domain level, we designed a Map function that associates key/value record pairs where the *key* is the URL of the single web page, and the *value* is the corresponding web domain. The Reduce function, in turn, fulfills the setup, launch and execution of a multi-corpora GATE application (each corpus containing text documents and pages belonging to a single web domain), as well the subsequent estimation of extracted keywords/keyphrases relevance at web domain level (as later described).

C. GATE Application

This functional block of the *Keywords/Keyphrases MapReduce Extractor* module integrates and executes the GATE application in the MapReduce environment, according to the input configuration parameters and the pipeline, defined in an external configuration .xgapp file. This XML-based file is defined, by extension, as the effective GATE application, containing file paths and references to all the Processing Resources and plugins used. In this specific case, the ANNIE (*A Nearly-New Information Extraction System*) plugin, more specifically the *Tokenizer* and the *Sentence Splitter* tools, have been used to parse and segment the text content of crawled documents, while the *TreeTagger* plugin has been used for POS-tagging. Finally, the Java written *JAPE* (*Java Annotation Pattern Engine*) plugin syntax has been employed to define custom rules for filtering undesired, noisy parts of speech (such as conjunctions, adverbs, prepositions etc.). These rules are contained in a dedicated .jape file. Common nouns and adjectives are then annotated as potential keywords candidates. Candidate keyphrases are then identified as contiguous phraseological combinations and patterns of candidate keywords.

[8] http://nutch.apache.org/
[9] http://lucene.apache.org/solr/

Regarding the execution of the GATE application in the Hadoop distributed environment, the following strategy has been followed: the Namenode loads, at run time, a zip archive containing all the needed GATE APIs, configuration and application files, libraries and plugins, in the HDFS Distributed Cache. This is one of the possible solutions adopted for handling read and write operations on files in the HDFS. This has been considered an efficient solution since, by this way, our application will copy in memory and extract the necessary files only once, and the allocated content will be accessible by all the Datanodes, without the need of installing required plugins and third party tools on each single cluster node. An additional advantage of this approach stems by the fact that any generic GATE application can be potentially executed in our Hadoop-based architecture (taking benefits of all the NLP features and capabilities offered), providing to embed the .xgapp application file, the .jape file for annotation rules and patterns, as well as all the required resources and plugins in the input zip file.

D. TF-IDF Relevance Estimation

Relevance estimation for candidate keywords and keyphrases is performed by computing the TF-IDF (*Term Frequency – Inverse Document Frequency*) function. This metric is widely adopted in Information Retrieval to assess how significant is a given term not really within the single document in which it has been retrieved, but rather with respect to the whole documents collection (corpus). Actually, a lot of common words like articles or conjunctions may appear several times in a document but they are not relevant as key-concepts to be indexed or retrieved. As a matter of fact, TF-IDF is given by the combination of two functions: TF (*Term Frequency*) which provides a measure about how frequently a term occurs in a certain document, and IDF (*Inverse Document Frequency*) which measures how important is a certain term with respect to the whole corpus. The final TF-IDF value for a candidate keyword *k* in a document *d* within a corpus *D* is calculated as:

$$(TF - IDF)_k = TF_k \cdot IDF_k \quad ,$$

where:

$$TF_k = \frac{f_k}{n_d} \quad , \qquad IDF_k = \log \frac{N_D}{N_k}$$

being f_k the number of occurrences of the candidate keyword *k* in the document *d*, n_d the total number of terms contained in document *d*, N_D the total number of documents in the corpus *D* and N_K the total number of documents within the corpus in which the candidate key *k* appears.

After calculating TF-IDF for each potential keyword and keyphrase, candidates with a TF-IDF value under a defined threshold are pruned, while those with a value above the threshold are designed as definitive keywords/keyphrases. These ones are finally stored in the Hadoop HDFS file system, annotated together with their corresponding TF-IDF values and source web domain URL.

E. DB Storage

This module accomplishes the storage of the final system output (temporarily stored in the HDFS by the Extractor Module) in an external SQL database. This operation is carried out by means of the Apache Sqoop [10] open source tool, specifically designed for data transfer between Hadoop HDFS and structured datastores. The Sqoop tool has been installed on the master only and the export feature has been used to insert data stored in the HDFS into the database. In order to successfully accomplish the data export, full read/write privileges on the database have been granted to all the machines on the cluster. Each database record is populated with an extracted keyword or keyphrase, its corresponding POS-tag (or a different custom tag if it is a keyphrase), TF-IDF value and the source web domain.

IV. EVALUATION

The performances of the proposed system have been evaluated against a dataset composed of 10000 web page and documents, which is a subset of the resources ingested by our Distributed Crawler module (which has currently been gathering more than 6 million web URLs). The Hadoop cluster architecture used for tests has been assessed on different configurations, ranging from 2 to 5 nodes. Each node is a Linux 8-cores workstation whith Hadoop HDFS installed. The master Namenode, besides, required also the Apache Sqoop software installation to manage SQL write operations for final output storage. In order to avoid data integrity errors and failures due to decommission and recommission of cluster nodes, Hadoop allows to perform a rebalance of stored blocks among the active nodes of the cluster, if necessary.

For each cluster configuration, a fixed number of keywords/keyphrase extraction tests has been performed, on the same defined dataset. The MapReduce model supplies speculative execution of tasks, and it is designed to provide redundancy in order to handle fault tolerance. By this way, it may happen that the Namenode JobTracker has to reschedule failed or killed tasks, and this can affect the execution time of the whole process. Therefore, for performance comparison, the best processing times have been selected among all the tests performed for each node configuration. By this way, the number of attempts for re-executing failed or killed tasks is supposed to be minimized. For the whole test dataset, containing about 10000 documents, a total of nearly 3.5 million keywords and keyphrases have been extracted. Time processing results for the different tested node configurations are shown in Table 1. The single-node configuration has been actually implemented as a two-nodes cluster, with a master Namenode and one slave node running only as a Datanode (in order to avoid HDFS space and blocks balancing problems on

[10] http://sqoop.apache.org/

decommissioning too many nodes). That is, the TaskTracker daemon stopped (so that it does not take part to the MapReduce process). As a term of comparison, running the same GATE application on the same corpora dataset on a single non-Hadoop workstation took approximately 60 hours. A possible explanation to this significant performance gap can be the fact that the Java code of our standalone GATE application is not optimized for multi-thread, while the MapReduce adaptation executed in Hadoop can benefit of MapReduce configuration parameters, which define the maximum number of map and reduce task slots to run simultaneously (exploiting multi-core technology even on a single-node cluster). The resulting speed-up curve for our test data is shown in Figure 2.

As it can be noticed, the scaling capabilities of the proposed system confirm the nearly linear growth trend of the Hadoop architecture. However, the low curve slope suggests that significant performance improvements can be achieved only for larger number of nodes in the cluster.

TABLE I. EVALUATION RESULTS: TIME PERFORMANCES ASSESSED FOR DIFFERENT CLUSTER CONFIGURATIONS.

Configuration	*Processing Time (hh:mm:ss)*	*Speed - Up*
HDFS - single node	07:17:01	-
HDFS - 2 nodes	05:21:53	1.36
HDFS - 3 nodes	04:08:00	1.76
HDFS - 4 nodes	03:39:42	1,99
HDFS - 5 nodes	03:20:09	2.18

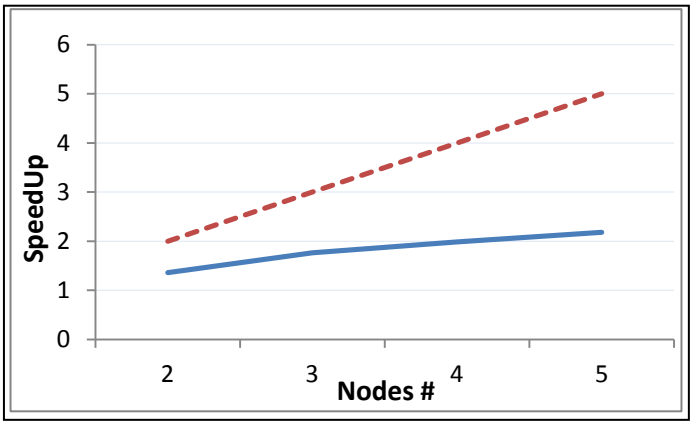

Figure 2. Processing time performances depicted for distributed extraction of Keywords and Keyphrases (solid curve), performed with different cluster configurations, against the ideal linear trend (dotted curve).

V. CONCLUSIONS AND FUTURE WORK

In this paper, a distributed system for keywords and keyphrases extraction from text content of web pages and documents has been presented. The parallel architecture is provided through the implementation of the open source Apache Hadoop framework, while text annotation and key features extraction rely on the NLP opens source GATE platform. The main advantages and contributions brought by the proposed system are two: the first is the integration of a web crawler which allow to use our system as a standalone application, avoiding interfacing issues with external tools for documents ingestion and indexing (actually, the most part of current NLP tools assumes that input documents are already collected, properly cleaned and formatted). In addition, the proposed system provides the capability of potentially executing any generic GATE application (thus allowing to perform a wide range of NLP activities) in a distributed design, without the need for programmers to modify and update every time the code, in order to follow the steps usually required for parallel computing development (such as task decomposition, mapping and synchronization issues). Open issues for future work are, in addition to test a wider range of GATE based applications, the evaluation of processing times on larger cluster configurations and larger document corpora, with the goal of better assessing the trend of performance improvements. Moreover, it could be interesting to adapt the Hadoop implementation of the keywords/keyphrases extraction on other parallel computing, distributed architecture, such as Spark. Regarding possible quality improvements to the currently adopted NLP solutions used for key features extraction, the annotation of proper nouns (for instance, VIP person names and toponyms) could also be provided, by the use of lists, gazetteers and other external knowledge resources. This might be useful, as well as enhancing the relevance of produced output, for content annotation, summarization and domain characterization purposes in higher expressive contexts, such as Semantic Computing frameworks.

REFERENCES

[1] J. Lin, and C. Dyer, “Data-Intensive Text Processing with MapReduce”, Morgan & Claypool Publishers, 2010.

[2] F. Colace, M. De Santo, L. Greco and P. Napoletano, “Text classification using a few labeled examples. Computers in Human Behavior”, Vol. 30, pp. 689-697, 2014.

[3] R. Al-Hashemi, “Text Summarization Extraction System (TSES) Using Extracted Keywords”, International Arab Journal of e-Technology, Vol. 1(4), pp. 164-168, 2010.

[4] F. Colace, M. De Santo, L. Greco and P. Napoletano: Weighted Word Pairs for query expansion. Inf. Process. Manage. 51(1): 179-193 (2015).

[5] O. Kononenko, O. Baysal, R. Holmes and M. W. Godfrey, “Mining Modern Repositories with Elasticsearch”, in Proc. of the 11th Working Conference on Mining Software Repositories, pp. 328-331, 2014.

[6] N. Ide. and L. Romary, “International standard for a linguistic annotation framework”, Natural Language Engineering, Vol. 10(3-4), pp. 211-225, 2004.

[7] A. Hulth, “Improved automatic keyword extraction given more linguistic knowledge”, in Proc. of the 2003 Conference on Emprical Methods in Natural Language Processing, Sapporo, Japan, 2003.

[8] T. Luis, “Parallelization of Natural Language Processing Algorithms on Distributed Systems”, Master Thesis, Information Systems and Computer Engineering, Instituto Superior Técnico, Univ. Técncica de Lisboa, 2008.

[9] T. Hamon, J. Deriviere and Nazarenko, “Ogmios: a scalable NLP platform for annotating large web document collections”, in Proc. of Corpus Linguistics, Birmingham, United Kingdom, 2007.

[10] J. Lin and C. Dyer, “Data-Intensive Text Processing with MapReduce“, Morgan & Claypool Publishers, 2010.

[11] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan, “GATE: A Framework and Graphical Development Enviroment for Robust NLP

Tools and Applications," in Proc. of the 40th Anniversary Meeting of the Association for Computational Linguistics, ACL '02, Philadelphia, 2002.

[12] C. Zhang, H. Wang, Y. Liu, D. Wu, Yi Liao and Bo Wang, "Automatic Keyword Extraction from Documents Using Conditional Random Fields", Journal of Computational Information Systems, 2008.

[13] Y. Matsuo and M. Ishizuka, "Keyword extraction from a single document using word co-ocuurrence statistical information", International Journal on Artificial Intelligence Tools, 2004.

[14] J. Kaur, V. Gupta, "Effective Approaches For Extraction Of Keywords", IJCSI International Journal of Computer Science Issues, Vol. 7(6), pp. 144-148, November 2010.

[15] I. Witten, G. Paynte, E. Frank, C. Gutwin and C. Nevill-Manning, "KEA: practical automatic keyphrase extraction", in Proc. of the 4th ACM Conference on Digital Library, 1999.

[16] C. Wu, M. Marches, J. Jiang, A. Ivanyukovich and Y. Liang, "Machine Learning-Based Keywords Extraction for Scientific Literature", Journal of Universal Computer Science, Vol. 13(10), pp. 1471-1483, 2007.

[17] Z. Liu, P. Li, Y. Zheng and M. Sun, "Clustering to find exemplar terms for keyphrase extraction", in Proc. of the 2009 Conf. on Empirical Methods in Natural LanguageProcessing, pp. 257–266, 2009.

[18] F. Liu, D. Pennell, F. Liu and Yang Liu, "Unsupervised approaches for automatic keyword extraction using meeting transcripts", in Proc. of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp. 620-628. 2009.

[19] O. Medelyan, E. Frank and I. H. Witten, "Human-competitive tagging using automatic keyphrase extraction" in Proc. of the 2009 Conf. on Empirical Methods in Natural Language Processing, pp. 1318-1327, 2009.

[20] K. S. Hasan and V. Ng, "Automatic Keyphrase Extraction: A Survey of the State of the Art", in Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics, Vol. 1, pp. 1262-1273, 2014.

[21] M. Grineva, M. Grinev and D. Lizorkin, "Extracting key terms from noisy and multitheme documents", in Proc. of the 18th Int. Conf. on World Wide Web, pp. 661-670,2009.

[22] Z. Liu, W. Huang, Y. Zheng and M. Sun, "Automatic keyphrase extraction via topic decomposition", in Proc. of the 2010 Conf. on Empirical Methods in Natural Language Processing, pp. 366-376, 2010.

[23] M. Chung and D. I. Moldovan, "Parallel Natural Language Processing on a Semantic Network Array Processor", IEEE Transactions on Knowledge and Data Engineering, Vol. 7(3), pp. 391-404, 1995.

[24] M. P. van Lohuizen, "Parallel Processing of Natural Language Parsers", in Proc. of the 15th Conf. of Parallel Computing, pages 17-20, 2000.

[25] P. Jindal, D. Roth and L.V Kale, "Efficient Development of Parallel NLP Applications", Tech. Report of IDEALS (Illinois Digital Environment for Access to Learning and Scholarship), 2013.

[26] N. Rizzolo and D. Roth, "Learning Based Java for Rapid Development of NLP Systems". In Proc. of the International Conference on Language Resources and Evaluation (LREC), 2010.

[27] L. V. Kale and G. Zheng, "Charm++ and AMPI: Adaptive Runtime Strategies via Migratable Objects", in Advanced Computational Infrastructures for Parallel and Distributed Applications, pp. 265-282, Wiley Interscience, 2009.

[28] Exner, P. and Nugues, P., "KOSHIK - A Large-scale Distributed Computing Framework for NLP", in Proc. of the International Conference on Pattern Recognition Applications and Methods (ICPRAM 2014), pp. 463-470, 2014.

[29] V. Tablan, R. I. Cunningham and K. Bontcheva, "GATECloud.net: a platform for large-scale, open-source text processing on the cloud", Philosophical Transactions of the Royal Society, 2013.

[30] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker and I. Stoica, "Spark: Cluster Computing withWorking Sets", Technology report of UC Berkeley, 2011.

[31] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma; M. McCauly; M. J. Franklin, S. Shenker and I. Stoica, "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing", in Proc. of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12), pp. 15-28, 2012.

[32] S. Gopalani and R. Arora, "Comparing Apache Spark and Map Reduce with Performance Analysis using K-Means", International Journal of Computer Applications ,Vol. 113(1), pp. 8-11, 2015.