A Distributed Text Mining System for Online Web Textual Data Analysis

Bin Zhou, Yan Jia
School of Computer
National University of Defense Technology
Changsha, Hunan 410073, China
e-mail: binzhou@nudt.edu.cn, jiayanjy@vip.sina.com

Chunyang Liu, Xu Zhang
Section of Information
CNCERT/CC
Beijing 100029, China
e-mail: zhangxu@isc.org.cn

***Abstract*—Real world Web mining applications usually have different requirements, such as massive data processing, low system latency, and high scalability. In order to meet these different requirements, we proposed a distributed text mining system with a layered architecture that divides the system functions into three layers, namely, the crawling and storage layer, the basic mining layer, and the analysis service layer. Message-oriented middleware are used between these layer components and services to make the communication in a loosely-coupled way. To conquer the data-intensive and storage failure problems, a distributed file system is used to store and manage the raw text data and various indexes. As a case study and example, the design and implementation of an experimental online topic detection application, which can be scaled to handle thousands of Internet news and forum channels and perform online analysis, is also discussed.**

Keywords- distributed computing; text mining; information discovery

I. INTRODUCTION

Web has become a very convenient and effective communication channels for people to share their knowledge, express their opinion, promote their products, or even educate each others, by publishing textual data through a browser interface. Mining meaningful and useful information from those plain textual data is important for people to uncover those hidden knowledge. Thus, online Web textual data analysis applications become popular. [1,2,3].

A more careful review of the online textual data analysis applications will find that most of these tasks have some common or similar requirements.

(1) *Different requirements co-exist in the system*. For example, most underlining basic text mining functions, e.g. annotation, stemming, word counting, Chinese word segmentation and named entity recognition, need to be run continuously (24 hours a day and 7 days a week), which are often task based or job based computation. However, other higher level applications, e.g. topic discovery or retrieval, are more likely to be exploratory and query based, which usually run in interactive ways with low latency requirements.

(2) *Data-intensive computation*. By data-intensive we mean that the data volume of data analysis task is very large in size (normally, the data volume is at GB and TB scale). In this case, to move the data is very expensive in cost. At the same time, disk error will be a common phenomenon when storing such a big volume of data.

(3) *Scalability is crucial*. The computational platform needs to be scalable in order to accommodate more future incoming data. Because the Web textual data keeps on arriving from different sources, it will be good to add more computing nodes without interrupting current analysis procedure.

In this paper, we introduce a distributed text mining system for online Web textual data analysis. In order to meet the different requirements on system latency, we propose a layered architecture to divide the whole system function into three layers, namely, the crawling and storage layer, the basic mining layer, and the analysis service layer. The functions of these layers will be discussed in later sections. Message-oriented middleware are used between these layers to make the communication in a loosely-coupled way. To conquer the data-intensive and storage failure problems, the Hadoop distributed file system [4] are used to store and manage the raw text data and the indexes. As a case study and example, we also discussed the design and implementation of an online topic detection application which can handle thousands of Internet forum channels and perform online analysis.

The rest of the paper is organized as follows. Section 2 provides an overview on existing text mining systems and frameworks. Section 3 introduces the distributed text mining system in detail. In Section 4 we describe a case study of topic detection for Chinese Internet forums. Section 5 describes the experiments conducted so far on different dataset to validate the effectiveness of the proposed methods. Section 6 concludes the paper and discusses future works.

II. RELATED WORKS

The Unstructured Information Management Architecture (UIMA) project from IBM Research aims to set up common software architecture for developing, composing and delivering unstructured information management technologies [5]. As a standard and reference, UIMA provides a set of interfaces for defining components for analyzing unstructured information and provides infrastructure for creating, configuring, running, debugging, and visualizing these components. The basic mining layer of our system is also built on UIMA. However, with regard to the problems previously mentioned, UIMA's main data type is text stream and does not take the GB/TB scale storage management into consideration.

978-0-7695-4235-5/10 $26.00 © 2010 IEEE
DOI 10.1109/CyberC.2010.11

Similar to our work, ClearTK [6] is another framework which is built on top of UIMA. It supports statistical NLP by providing a feature extraction library, and a set of components for NLP tasks such as tokenization, part-of-speech tagging, named entity identification, and semantic role labeling, etc. It also has interfaces to other machine learning libraries, such as Mallet [7], Weka [8], etc. Comparing to our work in this paper, ClearTK is more like a toolbox with integrated interfaces to other third-party NLP tools. However, we are focusing on the design of a distributed text mining system which can tackle Web text data analysis at TB level.

III. DISTRIBUTED TEXT MINING SYSTEM

A. Layered Architecture

In order to meet the requirements of different services, the whole system are divided into three layers. As illustrated in Figure 1. , from bottom up, they are the Crawling and Storage Layer (CS Layer), the Basic Mining Layer (BM Layer), and the Analysis Services Layer (AS Layer), respectively.

The main component of the CS layer is a distributed focused Web crawler which can fetch documents from different Web sources. Currently, the crawler supports many different types of information sources, such as Internet forums, news portal, blog sites, and micro-blogs. It runs in a 7×24 mode. The storage management of the CS layer is based on the Apache Hadoop Distributed File System (HDFS) [4]. Because HDFS supports file-block level replication across machines and cluster, our system is also highly fault-tolerant to random disk failures. Besides, HDFS is suitable for applications that have large data sets and is tuned to support large files, which meets the Web content analysis application well. However, after storing the massive amount of Web texts, HDFS is not enough for retrieving them efficiently on the requests from the upper levels. For real-time read/write file access purpose, we count on the HBase [9] for the management of the crawled documents and their meta-information, such as the resource types and timestamps etc.

The BM layer is built on top of the UIMA framework. In essence, UIMA framework is a language processing middleware which takes container-component architecture. All the user-defined procedures are implemented as UIMA components. There are mainly two kinds of components, text preprocessing components such as *detagging and extraction*, *Chinese word segmentation* and *English stemming*; and various NLP annotators such as *Named Entity Annotator*, *Session Annotator*, and *Opinion Annotator*; and more advanced NLP processors such as *Text Classification and Clustering*, *Topic Detection and Tracking* and *Author-Topic Mapping*. By defining a standard data structure, the Common Analysis Structure, CAS, both the row input documents, meta data and the annotation data produced by the user-defined processes are encapsulated into a single data object.

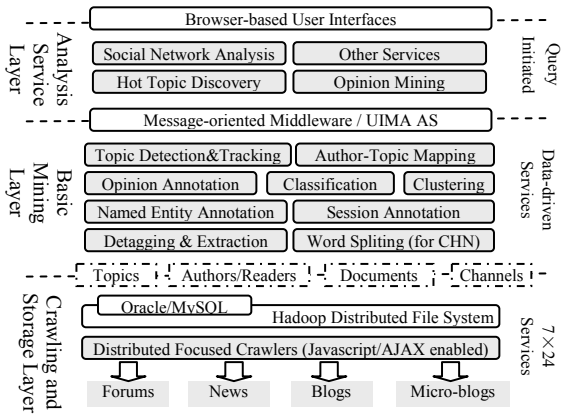

Figure 1. Layered architecture of the system.

The AS layer consists of the application specific services each of which normally solves certain type of user defined analysis services, such as *Hot Topic Discovery*, *Opinion Mining*, and *Social Networking Analysis*. This layer usually has a user interface to present the analysis output to the users and get the input from them. In this work, we use a browser based user interface to demonstrate the analysis results.

As we have mentioned in the previous section, the AS layer should be of low latency to meet with the interactive requirements of the applications. Meanwhile, the BM layer services can be more task-oriented, which do not put too much constraint on the system responsiveness. We used a very simple but effective method to solve this problem, i.e. setting up a structured data format (as database tables) to separate these two computational styles. All the BM layer services write their output into the database, and the AS layer applications read from the database tables to compose the presentation pages.

B. Asynchronized Data-driven Computation

The most important part which determines the system's performance of data analysis tasks lies in the BM layer. It is because in this layer, most of the basic mining functions are going to be carried out on every piece of data collected by the crawlers. In order to finish these basic tasks in an efficient way, we have used several concrete techniques and leverage their combinations to get better performance.

1) Data-driven workflow based on asynchronous communication between multiple instances. As the previous sub-section has suggested, BM layer is built on top of UIMA. In order to achieve high system throughput, we adopted the UIMA Asynchronous Scaleout (UIMA AS) [5] technique to enable the asynchronous communication between different UIMA instances. The main idea is the use of message-oriented middleware (JMS) to facilitate the communication between different computing nodes. UIMA services enable clients to utilize analysis engines running in separate processes on the same or different machines. In this way, the time-consuming computation in various BM layer services can be scattered to multiple instances across different machines, without worrying about the communication between them.

Figure 2. gives an example workflow of the BM layer services. The whole workflow was started by polling on the Hbase for the newly incoming Web text files collected by the crawlers. Once the newly documents are located, their URLs are added to the message queues (Step①). Following a Pub-Sub paradigm, the message will be pushed to the UIMA instances when they are available (Step②). The following processes can be deployed on the same machine or on different machines (Step③-④). After the procedures are finished, the responses messages will be put back to the message queues (Step⑤), which is written into the database tables (Step⑥). Roughly, the message queues can be divided in two categories, the requesting message queues and response message queues. All the queues can have different priorities.

2) Map-Reduce programming model to move computation closer to data.

Another important feature of our system is that the introduction of MapReduce programming paradigm. As illustrated in Figure 2. , some time consuming tasks (e.g. term frequency counting) can be decomposed into many computing nodes following the MapReduce programming framework. This can help to dispatch and move computation closer to the data where they are stored. At the same time, the process of term frequency counting is just a single logic step of the workflow. This can simplify the modeling very much.

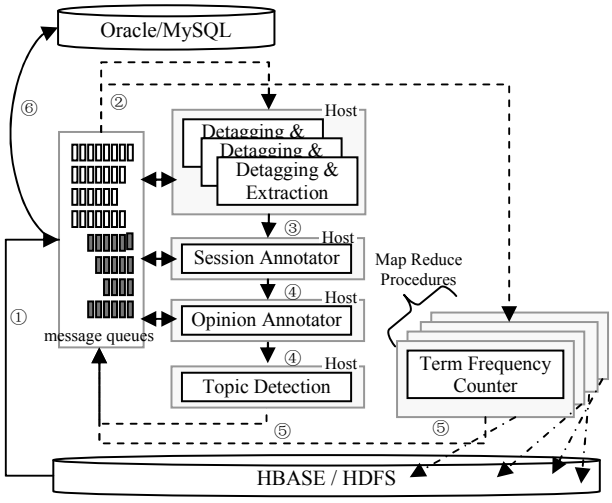

Figure 2. An example workflow of BM layer services.

IV. A CASE STUDY: TOPIC DISCOVERY FROM INTERNET FORUMS

A case study of topic discovery from Internet forums is performed to verify the functions of the distributed text mining system.

A. Basic Mining Services for Topic Discovery

There are several kinds of BM services for topic discovery. Among them, the *Information Extractor* (IE) and *Chinese Word Segmentation* (CWS) services are the most important ones. They are all implemented as UIMA Analysis Engines (AEs). The IE service deals with the extraction of the targeting text snippets such as document title, creation time, last modified time, user-defined tags (if any), author, reply message count, etc. Normally, different forums and channels have different segments, so the extractors have to be carefully configured and managed for the IE service to be robust.

After the targeting text snippets have been identified and collected, it is time for extract language features from them. Most of our algorithms used so far are based on word bag model. The first step is Chinese word segmentation. The CWS service is responsible for this purpose. We use ICTCLAS [10] as the segmenter and wrapped it as an UIMA AE.

B. CAS elements for Topic Discovery

Common Analysis Structure (CAS) is the core of the BM and AS services. By exploring a responsibility chain design pattern [11], all the AEs collaborate with each other by passing the shared CAS objects and carrying on their own operations on them one by one.

For example, as an earlier step, IE read the raw text, which may be a forum post, from the CAS which is created by a Collection Reader at earlier step of the chain. Then, it extracts the targeting snippets according to different extraction templates it is configured and writes the extracted snippets back to the same CAS object, before passing it to the next AE, say, the CWS. Activated by the *next()* operation of IE, CWS can perform its own task, word segmentation, on the snippets fetched from the CAS it received.

V. EXPERIMENTS

Preliminary experiments have been done to test the function and performance of the system proposed in this paper. We set up a test environment of 11 hosts, 3 nodes for Web crawlers, 4 nodes for the HBase and HDFS storage, 3 nodes for mining services, and a node for the database server. We have prepared a dataset composing of more than one million documents collected from different channels of Chinese Web news, forum posts, and micro-blogs. The total data size is more than 70GB. These documents are all stored as tuples in the big table of Hbase.

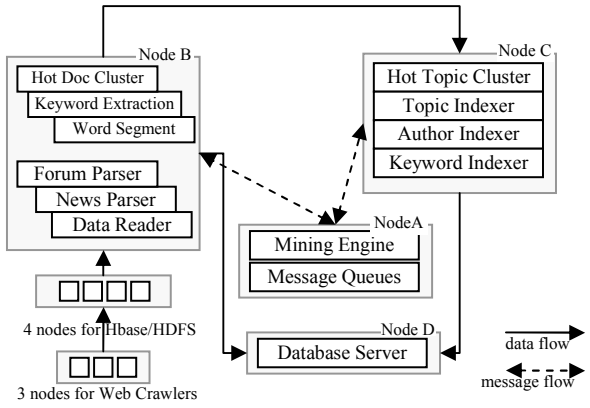

Figure 3. Experimental setup.

Figure 3. is the experimental setup. Note that node A is the main controller of the experiments which runs the UIMA AS engine and the ActiveQ message queues. Node B and C are the main service runners. The raw text data are loaded from the Hbase by component *Data Reader*. The title, publication datetime, author, content, visits, reply and comments are extracted from the raw data by component *News Parser* and *Forum Parser*.

To evaluate the performance of the services (implemented as UIMA AEs), we run every service as one single AE instance, and measure the time consumption of each instance. Figure 4. gives a test result of the time costs of the main mining services of the system. Note that most of the component's performances are linear to document numbers, except component for hot topic discovery and hot term discovery. This can be explain that the two processes are all global processes that they need global information to evaluate which topic (term) is hot comparing to its history.

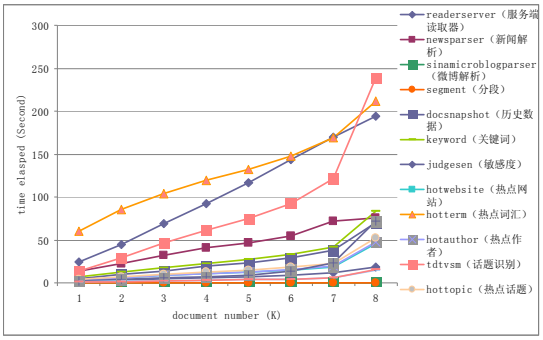

Figure 4. Performance of every service as one single AE instance.

In order to solve the scalability problem of the massive data processing task, we make use of the built-in scalability of the UIMA AS. To verify its scalability, we also take two components, namely the author indexer and the topic indexer, as examples. Two series of tests are performed. We use 1, 2, and 4 UIMA instance(s) for each component respectively, and calculate the speedup as a measure to see the scalability of the system. Figure 5. gives the speedup of the experiment. It is easy to see that the speedup is close to 2, which meet our expectation well.

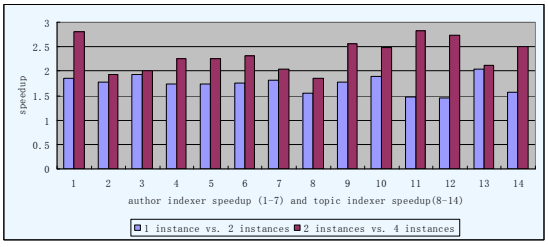

Figure 5. Speedup of two components.

Examplar applications were developed and verified the system has acceptable system latency. This is because that by separation the system into three layers, the application only need to contact with the database server and the indices produced by the mining services.

VI. CONCLUSIONS

In this paper, we propose a distributed text mining system with a layered architecture that divides the system functions into three layers. To conquer the data-intensive and storage failure problems, a distributed file system is used to store and manage the raw text data and various indexes. As a case study and example, the design and implementation of an experimental online topic detection application is also discussed.

The proposed method is still a preliminary work towards massive Web text processing. It can be extended in many directions. For example, whether the system can be extended to handle TB scale Web data is still unknown. We are also interested in developing a monitoring mechanism to monitor and adjust the system configuration automatically.

ACKNOWLEDGMENT

The authors would like to thank Zihai Jiang, Dafu Shan, and Zheng Liang for their helpful discussions and data preparations. The demo application is developed by Rui Zeng, Haiwen Chen, and Kun Peng. The work in this paper is supported by NSFC under the grant 60873204 and 60933005. It is also partially supported by the National 242 program under the grant 2009A90.

REFERENCES

[1] Qiaozhu Mei and ChengXiang Zhai, "Discovering evolutionary theme patterns from text: an exploration oftemporal text mining", In SIGKDD, pages 198-207, New York, NY, USA, 2005. ACM.

[2] ChengXiang Zhai, Statistical Language Models for Information Retrieval: A Critical Review, Foundations and Trends in Information Retrieval, Vol. 2, No. 3 (2008), pages 137-215, doi:10.1561/1500000008.

[3] Deng Cai, Qiaozhu Mei, Jiawei Han, ChengXiang Zhai, Modeling Hidden Topics on Document Manifold , Proceedings of the 17th ACM International Conference on Information and Knowledge Management (CIKM'08), pages 911-920.

[4] D. Borthakur. The Hadoop Distributed File System: Architecture and design. Hadoop Project Website, 2007.

[5] Apache UIMA Website, http://uima.apache.org/.

[6] ClearTK Website, http://code.google.com/p/cleartk/.

[7] McCallum, Andrew Kachites. "MALLET: A Machine Learning for Language Toolkit." http://mallet.cs.umass.edu. 2002.

[8] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.

[9] Hbase Website, http://hadoop.apache.org/hbase/.

[10] Hua-Ping Zhang , Hong-Kui Yu , De-Yi Xiong , Qun Liu, HHMM-based Chinese lexical analyzer ICTCLAS, Proceedings of the second SIGHAN workshop on Chinese language processing, p.184-187, July 11-12, 2003, Sapporo, Japan

[11] Gamma E., Helm R., Johnson R., Vlissides J. (1995) Design patterns: Elements of reusable -object-oriented software. Addison Wesley, Reading, MA