Performance Analysis of Apache Hadoop for Generating Candidates of Acronym and Expansion Pairs and Their Numerical Features

Taufik Fuadi Abidin
Department of Informatics
Syiah Kuala University
Banda Aceh, Indonesia
taufik.abidin@unsyiah.ac.id

Ramzi Adriman
Dept. of Electrical and Computer Engineering
Syiah Kuala University
Banda Aceh, Indonesia
ramzi.adriman@unsyiah.ac.id

Ridha Ferdhiana
Department of Statistics
Syiah Kuala University
Banda Aceh, Indonesia
ridha.ferdhiana@unsyiah.ac.id

***Abstract*—Mining information from web pages to automatically determine the acronym and its expansion is an important and challenging task. It is considered not easy to do because the acronyms writing rules and forms are very diverse for each language. The task consists of several important steps i.e. generating candidates of an acronym and its expansion, creating their numerical features, and determining the correct acronyms and expansions using machine learning algorithm. In this work, we evaluate and compare the performance analysis, in terms of speed, to generate the candidates of acronym and expansion pairs and create their numerical features when the processes are running on Hadoop cluster with different data nodes and on a single machine. The results show that Hadoop cluster outperforms a single machine when generating almost 52 million candidates of acronym and expansion pair and their numerical features. When Hadoop cluster was set up from two to three data nodes, the performance improved on average by 65.81%.**

***Index Terms*—Performance analysis, acronym and expansion, numerical features, Apache Hadoop**

I. Introduction

Data volume is increasing very rapidly nowadays. Humans, sensors, satellites, and other smart devices can now easily create various types of complex digital data, both structured and unstructured, such as texts, images, audios, and videos [1]. Handling this sheer volume of data, referred to as big data, using traditional computing paradigms and techniques suffer from the curse of cardinality because of its computational complexities. Moreover, the traditional computing techniques and platforms are less efficient, lack of scalability, and performance [2]. Therefore, one of the solutions is to process the data using a distributed and parallel computing paradigm that utilizes a cluster of computers and MapReduce computing paradigm [3].

Apache Hadoop is a framework that enables distributed computing across clusters of computers to process an enormous volume of data quickly [4]. As mentioned in the official website at http://hadoop.apache.org, the framework is designed to scale up to thousands of computers on a low-cost hardware, each offering local storage and computation power. The clusters can dynamically grow by adding more computers or virtual machines. Hadoop implements a special file system called Hadoop Distributed File System (HDFS) and uses map() and reduce() functions, commonly written in Java [5]. The map() function takes an input key-value pair and produces a list of intermediate key-value pairs. Afterward, the system will group all intermediate key-value pairs and pass them to the reduce() function for final results [5]. To accommodate codes written using a language other than Java, Hadoop Streaming library must be used.

YAVA is the distribution of Apache Hadoop that uses the power of Apache Hadoop ecosystem. It is an open source compilation of big data platform developed and maintained by a team at Labs247 as mentioned in the official website at http://yava.labs247.id/. YAVA implements HDFS, YARN cluster management, Apache Zookeeper, Apache Ambari to easily manage and monitor Hadoop clusters, Apache Sqoop, and many others modules. YAVA is intentionally compiled and bundled to accelerate the adoption of Hadoop technology in business and academia.

In the last few years, more and more researchers were working on big data using Hadoop technology. In 2014, Abuin et al. in [6] introduced Perldoop, a tool that can automatically translate Hadoop-ready Perl scripts into Java codes. They tested the tool using hundreds of regular expressions in a Hadoop cluster with 64 nodes and found that Java codes generated using Perldoop execute up to 12 times faster than the original Perl scripts that use Hadoop Streaming library. The work reported by Luna et al. in [7] proposed a series of algorithms for Apriori for pattern mining based on the MapReduce framework. They evaluated the performance of the proposed algorithms using up to $3 \cdot 10^{18}$ transactions and more than 5 million of distinct single-items and revealed the interest of applying MapReduce coding paradigm for big data analysis. Xun et al. in [8] proposed a parallel frequent itemsets mining method, referred to as FiDoop, which utilizes the MapReduce programming model in its implementation. The mappers decomposed itemsets and the reducers performed combination operations. Hadoop related work reported by Zhonghua in [9] evaluated the performance of extracting seismic data attributes as one of the solutions to the problem of big data in petroleum exploration applications. In addition,

978-1-5386-7082-8/18/$31.00 ©2018 IEEE

Yang and Wu in [10] conducted a parallel video data analysis based on Apache Spark, an analytics engine for big data processing, and experimented with an extensive video data analysis on some benchmark datasets. They found that the parallel processing framework is able to efficiently reduce the processing time for video action detection and near-duplicate video retrieval problems.

In this paper, we evaluate the performance analysis of Apache Hadoop to generate the candidates of acronym and expansion pairs and generate their numerical features. We compare the speed when the process is running on YAVA Hadoop parallel processing environment with different data nodes and a single machine. To generate the candidates of acronym and expansion pairs, we split the texts into several sentences based on delimiters such as period, comma, exclamation mark, question mark, and end-of-file. The sentences are further tokenized into words. A candidate of an acronym is estimated using a simple function, threshold, and dictionary. If the score of the function is below a given threshold, then the word is not an acronym and it will be ignored. Otherwise, the adjacent words on the left and right sides of the acronym are collected as the candidates of the expansion. Therefore, an acronym can have more than one candidates of expansions. The numerical features of the acronym and its expansion pair are created subsequently. The features consist of 8 numerical values. Our contributions are two-fold:

1) We implement codes in Perl as shown in Fig. 4 and run it using Hadoop Streaming to create a map job, submit the job to the Hadoop cluster, monitor the job until it ends, and then create a reduce job.
2) We evaluate performance analysis in terms of speed to generate the candidates of an acronym and its expansion and create their numerical features.

This paper is organized as follows. Section II describes the YAVA system setting for our Apache Hadoop cluster. Section III gives an overview of the methodology. Section IV discusses the performance evaluation, and Section V concludes the work.

II. YAVA System Setting

YAVA is an open source Hadoop distribution package that comes with a resource management layer, referred to as YARN

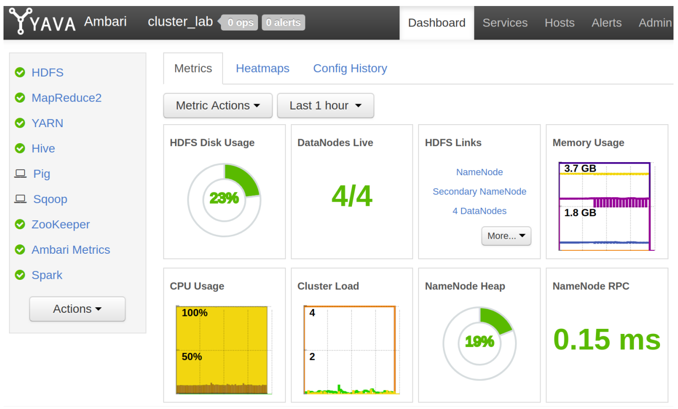


Fig. 1. Ambari interface

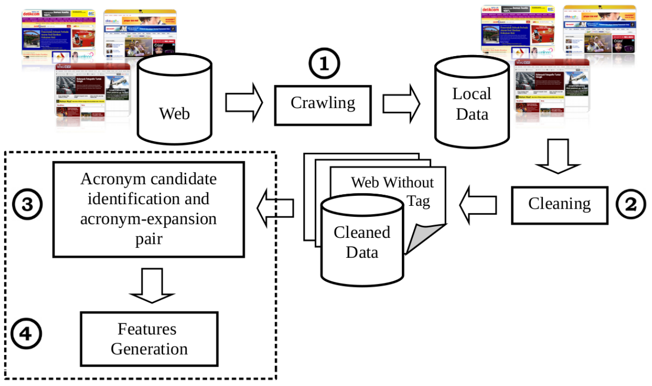


Fig. 2. The work flow, numbered into 4 major tasks

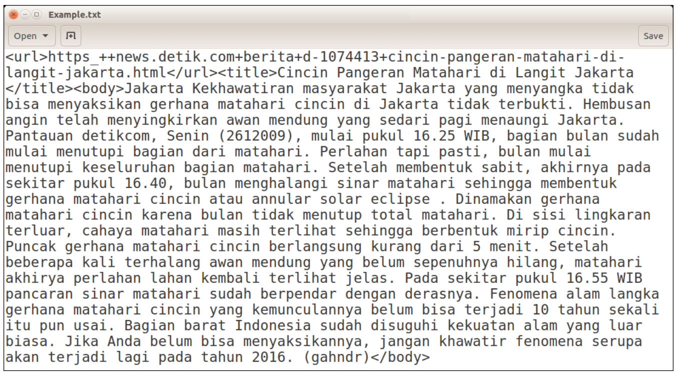
(Yet Another Resource Negotiator) [11], that allows various processes (interactive, batch, and streaming) to run on top of HDFS. YAVA allows smooth data transfer between Hadoop and database management system with Apache Sqoop. YAVA also comes with Apache Ambari that manages and monitors the cluster resources. It has a nice graphical user interface that can be integrated easily with other tools as depicted in Fig. 1, while Zookeeper stores and manages critical changes in the configurations and provides a distributed configuration and synchronization.

We setup an Ambari Server on a virtual machine with an IP public for easy access and monitor. The Ambari Server controls each host using Ambari Agent, setup in each node. We have 4 hosts setup for our cluster, each is labeled as tfa1 (192.168.1.211), tfa2 (192.168.1.212), tfa3 (192.168.1.213), and tfa4 (192.168.1.214) respectively. Host tfa1 is the virtual machine where Ambari Server is setup. Furthermore, each host has Centos7 (x86_64) OS installed, 4 CPU cores, 96.06 GB of storage, and 3.70 GB of RAM.

III. Methodology

In this section, we describe the methodology of the work as depicted in Fig. 2. There are four major workflows (1) data crawling, (2) data cleaning, (3) finding the candidates of an acronym and generating its expansion, and (4) creating the numerical features of the pairs.

We crawled 60,000 web pages from news.detik.com and cleaned them. Data cleaning includes removing HTML tags and specific symbols such as the hexadecimal numeral system to represent HTML color code, HTML entity name, e.g., "&copy;" for copyright sign, "&trade;" for trademark sign, "&nbsp;" for non-breaking space, and many others. We used HTML::ExtractContent Perl module to extract web page content [10] and utilized Perl regex to handle the specific symbols. The cleaned pages will only contain texts that are found in the title and body sections. An example of the page, downloaded from https://news.detik.com/berita/d-1074413/cincin-pangeran-matahari-di-langit-jakarta is shown in Fig. 3. The texts enclosed between <url> and </url> are the source URL, the texts enclosed between <title> and



```
<url>https_++news.detik.com+berita+d-1074413+cincin-pangeran-matahari-di-
langit-jakarta.html</url><title>Cincin Pangeran Matahari di Langit Jakarta
</title><body>Jakarta Kekhawatiran masyarakat Jakarta yang menyangka tidak
bisa menyaksikan gerhana matahari cincin di Jakarta tidak terbukti. Hembusan
angin telah menyingkirkan awan mendung yang sedari pagi menaungi Jakarta.
Pantauan detikcom, Senin (2612009), mulai pukul 16.25 WIB, bagian bulan sudah
mulai menutupi bagian dari matahari. Perlahan tapi pasti, bulan mulai
menutupi keseluruhan bagian matahari. Setelah membentuk sabit, akhirnya pada
sekitar pukul 16.40, bulan menghalangi sinar matahari sehingga membentuk
gerhana matahari cincin atau annular solar eclipse . Dinamakan gerhana
matahari cincin karena bulan tidak menutup total matahari. Di sisi lingkaran
terluar, cahaya matahari masih terlihat sehingga berbentuk mirip cincin.
Puncak gerhana matahari cincin berlangsung kurang dari 5 menit. Setelah
beberapa kali terhalang awan mendung yang belum sepenuhnya hilang, matahari
akhirya perlahan lahan kembali terlihat jelas. Pada sekitar pukul 16.55 WIB
pancaran sinar matahari sudah berpendar dengan derasnya. Fenomena alam langka
gerhana matahari cincin yang kemunculannya belum bisa terjadi 10 tahun sekali
itu pun usai. Bagian barat Indonesia sudah disuguhi kekuatan alam yang luar
biasa. Jika Anda belum bisa menyaksikannya, jangan khawatir fenomena serupa
akan terjadi lagi pada tahun 2016. (gahndr)</body>
```

Fig. 3. Example of a clean web page with three special tags

</title> are the title, and those enclosed between <content> and </content> tags are the content.

The process of finding an acronym and then generating the candidates of acronym and expansion pairs are handled by a subroutine called getCandidate() with the following subtasks:

1) Split the texts into sentences.
2) For each sentence, fetch each word and double check if the word is found in the dictionary. Compute the score of the word if it is not listed in the dictionary. The score is the ratio of total capital letters and the number of letters in that word.
3) If the score of the word is bigger than a given threshold, i.e. 75%, then the candidates of the expansion are generated by taking the neighboring words in the left and right sides of the acronym. Otherwise, ignore the word and process the next one in the sentence.

The subsequent task is to generate the features of the acronym and expansion pairs. The subroutine to handle this task is getFeature(). The subroutine will create the following eight numerical features:

- F_1 describes the relationship between the number of characters in the acronym and the number of words that are in the expansion. Typically, the number of letters in the acronym is the with the number of words in the expansion. Thus, feature F_1 will be equal to 1 if the number of letters in the acronym and the number of words in the expansion are the same. Otherwise, the value of F_1 will be less than 1.
- F_2 counts the first letter of each word in the expansion that is written in uppercase.
- F_3 shows the relationship between the acronym letters and the letters in the expansion. Acronyms are abbreviated based on the letters in their expansion, hence F_3 shows the similarity between the letters in the acronym and letters in the expansion.
- F_4 shows the relationship between the first letter of the acronym and the first letter of the first word in the expansion and the relationship between the last letter of the acronym and the first letter in the last word of the expansion. This feature is specifically designed to handle acronyms that are not entirely formed by the letters of each expansion word, for example, LP2M, which contains a digit in the acronym.
- F_5 illustrates that the expansion of an acronym usually does not contain a lot of prepositions and conjunctions. With this feature, the expansion that contains a lot of prepositions and conjunctions can be penalized as it is usually not the correct expansion of an acronym.
- F_6 is the ratio of the appearance of an acronym characters in its expansion.
- F_7 is introduced to have the difference between the correct acronym-expansion pair with the wrong one. The value of F_7 is 1 if $F_6 = 1$ and is 0 if $F_6 < 1$.
- F_8 is the average of the last seven numerical features.

The following examples show the candidates of acronym and expansion pairs with their numerical features. The last two pairs are examples of incorrect acronym and expansion pairs, whereas the other pairs show the correct acronym and expansion pairs with their corresponding features.

- *KPK:Komisi Pemberantasan Korupsi*
  $1, 1, 1, 1, 1, 1, 1, 1$
- *KBRI:Kedutaan Besar Republik Indonesia*
  $1, 1, 1, 1, 1, 1, 1, 1$
- *Menhub:Menteri Perhubungan*
  $1, 1, 1, 1, 1, 1, 1, 1$
- *Sumut:Sumatera Utara*
  $1, 1, 0.7, 1, 1, 1, 1, 0.96$
- *Bappenas:Badan Perencanaan Pembangunan Nasional*
  $0.99, 1, 0.4, 1, 1, 1, 1, 0.91$
- *Bappenas:Badan Perencanaan Pembangunan*
  $1, 1, -0.17, 0.5, 1, 0.88, 0, 0.60$
- *KBRI:Kedutaan Besar Republik*
  $0.99, 1, 0.67, 0.5, 1, 0.75, 0, 0.70$

The code as the mapper, written in Perl, is shown in Fig. 4. The codes use three important whitelists i.e. the list of stopwords, the list of basic words, and the list of conjunctions. In the codes, all whitelists are loaded into hashes for a quick lookup. Hadoop Streaming creates a map job of all statements in the while loop and submits it to the Hadoop cluster. It reads a non-empty line, takes the source URL, gets the content and cleans it. The getCandidate() function then generates candidates of acronym and expansion pairs and passes them into the getFeature() function. The function generates the eight numerical features of the acronym and expansion pairs. Last, Hadoop Streaming creates a reduce job. We run the codes with the following command:

```
hadoop jar /usr/yava/2.2.0.5/ \
  hadoop-mapreduce/hadoop-streaming- \
  2.7.1_yava_2.2.0.5.jar \
  -input /user/root/data.dat \
  -output /user/root/output \
  -mapper extracttfaMap.pl \
  -file extracttfaMap.pl \
  -file kata-dasar.txt \
  kata-sambung.txt stopword.txt
```

```
#!/usr/bin/perl -w
use Text::Ngrams;
use List::Util qw( max );
my $stopword = "stopword.txt";
my $katadasar = "kata-dasar.txt";
my $katasambung = "kata-sambung.txt";
my %katadasar;
my @katasambung;
my %stopword;
my @stopWords;
load_dict(\%katadasar);
load_katasambung(\@katasambung);
load_stop(\%stopword, \@stopWords);

while(<STDIN>){
  chomp;
  next if /^$/;
  $_ =~ m/<url>(.*?)<\/url><title>(.*?)$/;
  my ($url,$body)=($1,$2);
  my $c_url = $url;
  $c_url =~ s/_/:/;
  $c_url =~ s/\+/\//g;
  $c_url =~ s/\.ht\w+$//;
  my $clean_body = getCleanText($body);
  my $candidates = getCandidate($clean_body);
  my $features = getFeature($c_url,$candidates);
  print "$features";
}
```

Fig. 4. Perl codes for extracting candidates of acronym and expansion pairs and their numerical features

IV. Performance Evaluation

We compared the performance in terms of speed. The number of acronym and expansion pairs generated from 60,000 cleaned web pages are close to 52 million. In the performance evaluation, the speed of YAVA Hadoop cluster with different data nodes was compared with a single machine as depicted in Fig. 5. It can be seen clearly from the figure that the jobs running on Hadoop with three data nodes are faster than the two data nodes and single machine. The trend also indicates that Hadoop can have more processing power when more data nodes are added to the cluster. In addition, Hadoop is faster when processing a large amount of data while there is no significant difference in time when processing a small amount of data. Fig. 6 shows the number of candidates of acronym and expansion pairs generated from the cleaned web files.

V. Conclusion

MapReduce programming model and local computation power of data nodes in Apache Hadoop cluster can increase the processing speed of the cluster, and thus, the issue of scalability in big data can be solved. Our experimental results show that the time to generate almost 52 million candidates of acronym and expansion pairs and their numerical features using Hadoop distributed and parallel processing is faster than a single machine. The result also shows that the speed improved on average by 65.81% when a data node is added from two to three.

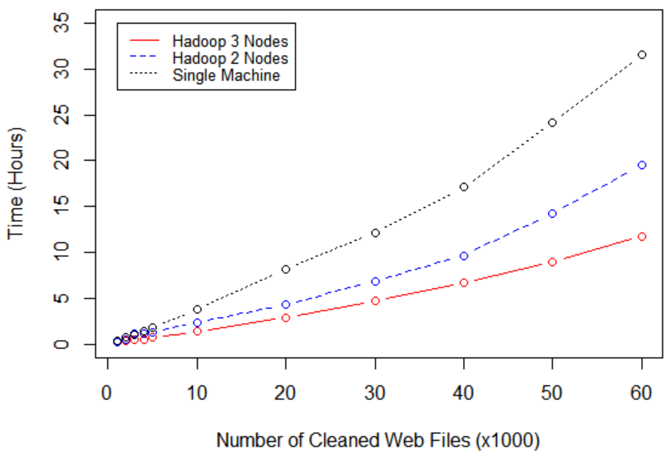


Fig. 5. Time comparison between three different settings

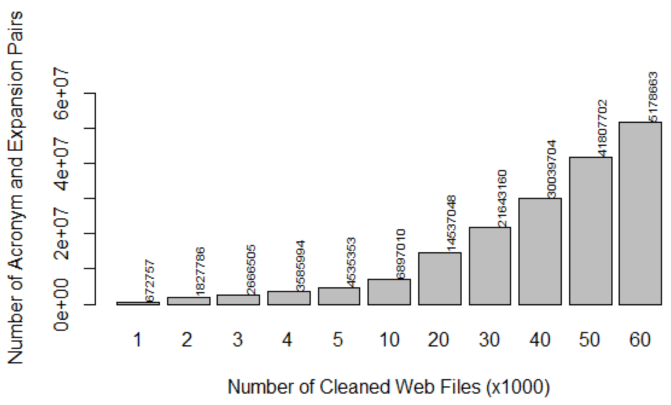


Fig. 6. Number of acronym and expansion pairs created per dataset

VI. Acknowledgment

The work is supported by Syiah Kuala University through H-Indexs Research Grant, contract number 42/UN11.2/PP/PNBP/SP3/2018. We would like to thank Sigit Prasetyo and Nur Kholis Majid, the technical experts from Solusi247, Syamsul Kamal and Steven Liunardi, the members of Data Mining and Information Retrieval Research Group, Department of Informatics, Syiah Kuala University for their technical supports and valuable thoughts.

References

[1] M. Chen, S. Mao, Y. Liu, “Big data: a survey,” Mobile Networks and Applications, vol. 19(2), pp. 171–209, 2014.

[2] A. Oussous, F. Benjelloun, A. A. Lahcen, S. Belfkih, “Big data technologies: a survey,” Journal of King Saud University – Computer and Information Sciences, vol. 30, pp. 431–448, 2018.

[3] J. Dean, S. Ghemawat, “MapReduce: simplified data processing on large clusters,” Communications of the ACM, vol. 51(1), pp. 1–13, 2004.

[4] M. R. Ghazi and D. Gangodkar, “Hadoop, mapreduce and HDFS: a developers perspective,” Procedia Computer Science, 48(C), pp. 45–50, 2015.

[5] R. Li, H. Hu, H. Li, Y. Wu, J. Yang, “MapReduce parallel programming model: a state-of-the-art rurvey,” International Journal of Parallel Programming, vol 44(4), pp. 83–866, August 2016.

[6] J. M. Abuin et al., “Perldoop: efficient execution of Perl scripts on hadoop clusters,” Proc. of the IEEE International Conference on Big Data, pp. 766–771, 2014.

[7] J. M. Luna, F. Padillo, M. Pechenizkiy, and S. Ventura, “Apriori versions based on MapReduce for mining frequent patterns on big data,” IEEE Transactions on Cybernetics, vol. 47(11), pp. 1–15, 2017.

[8] Y. Xun, J. Zhang, and X. Qin, “FiDoop: parallel mining of frequent itemsets using MapReduce,” IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 46(3), pp. 313–325, 2016.

[9] M. Zhonghua, “Seismic data attribute extraction based on Hadoop platform,” Proc. of the 2nd IEEE International Conference on Cloud Computing and Big Data Analysis, pp. 180–184, 2017.

[10] S. Yang and B. Wu, “Large scale video data analysis based on Spark,” Proc. of the International Conference on Cloud Computing and Big Data, pp 20–212, 2015.

[11] V. K. Vavilapalli, A. C. Murthy, C. Douglas, et.al, “Apache Hadoop YARN: yet another resource negotiator,” ACM Proc. of the 4th Annual Symposium on Cloud Computing, October 2013.