College Students-Oriented Internet Public Opinion System

Xi WU\textsuperscript{1}, Qiang XU\textsuperscript{2}, Ming ZHANG\textsuperscript{3}, Dan LIAO\textsuperscript{2} and Xiong WANG\textsuperscript{2}

\textsuperscript{1}Urban Vocational College of Sichuan, Chengdu 610000, China
\textsuperscript{2}School of Communication & Information Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
\textsuperscript{3}Chengdu Research Institute of UESTC, Chengdu 611731, China

Keywords: Distributed web crawler, Feature extraction, Text clustering, Parallelization.

Abstract. To improve the Internet public opinion monitoring for colleges, this paper designs an efficient Internet public opinion system for college students. The system uses distributed web crawler to collect the structuring information from the news websites, social networking sites, BBS and blogs, which are widely visited by college students. To speed up the text clustering process, the Single-Pass algorithm is parallelized with Spark. The parallel Single-Pass algorithm is used to find hot topics from collected texts. The algorithm improved the clustering efficiency significantly through distributed processing. Simulation results reveal that the proposed system can achieve timely and accurately public opinion monitoring.

Introduction

Internet public opinion refers to different views about current social problem on Internet, and is a form of social public opinion. In recent years, there has been frequent occurrence of government confidence crisis because of the lack of timely adequate response to Internet public opinion. As a group with high usage rate of Internet, college students are the main producers and transmitters of Internet public opinion. In order to prevent the rapid spread and deterioration of bad Internet public opinion, the department concerned must collect the Internet public opinion information as quickly as possible and keep track of their developments to supervise them effectively. So the Internet public opinion system came into being, which is an effective way to solve the problems above via information collection and topic detection. In order to improve the real-time performance of information collection and adapt to the current rapid growth of data, instead of traditional stand-alone web crawler, distributed web crawler is used for efficient and rapid information collection. On the other hand, text clustering algorithm can aggregate a large amount of text collected into a few meaningful clusters and detect hot topics. But traditional serial text clustering algorithms are inefficient when dealing with massive or high-dimensional data. A number of techniques have been developed to adapt text clustering algorithms to work with large datasets: examples are parallel programming frameworks such as MapReduce \cite{1} and cluster computing system such as Spark \cite{2,3}.

To summarize, this paper uses distributed technology to optimize and enhance the Internet public opinion system faced with big data. The distributed web crawler is based on an in-memory data structure store named Redis \cite{4}. And parallel text clustering algorithm is based on the Single-Pass clustering algorithm to generate clusters and on Spark framework to make parallel in-memory clustering tasks.

Distributed Web Crawler

Web crawler is a computer program used to crawl the source code of web pages and its basic workflow is as follows:

1) Set the initial webpage’s URL seeds to waiting queue.
2) Get a URL link from waiting queue. Add this URL link to filtering queue and download the webpage via this URL link if it isn’t included in filtering queue.

3) Parse webpage and extract all URL links and the main information in this webpage. Add URL links to waiting queue and webpage information to database.

4) Keep crawling until waiting queue is empty.

The workflow above is a basic design for web crawler. Faced with the huge data on Internet, the design of web crawler needs to be constantly optimized for different scenarios [5]. The system devote to crawl structured information from the news websites, social networking sites, BBS and blogs, which are widely visited by college students, such as Weibo, college BBS and so on. Often, a stand-alone web crawler is used to continuously crawl data on a website, but massive amounts of data across the website may become a performance bottleneck on account of limited computing power and network bandwidth. Distributed web crawler consist of multiple web crawlers deployed on different servers and it make the following improvements on the basis of a stand-alone crawler:

1) A public waiting queue is used to add all URL links extracted by all crawlers, and each crawler also gets URL link from this public queue.

2) A public filtering queue is used for deduplication. Before storing URL links to waiting queue, each crawler needs to remove duplicate URL links by the public filtering queue. Every new URL link should be added to filtering queue.

The distributed web crawler architecture diagram is shown as follows:

![Distributed web crawler architecture diagram](image)

In above architecture, multiple servers are used for crawler deployment. The Redis service deployed on one server is used as the public waiting queue and filtering queue. The MongoDB [6] service deployed on one server is used as database to store the main information of webpages. One crawler for one website is deployed on multiple servers so that even if a server is down the crawling for this website can still proceed smoothly.
Parallel Single-Pass Text Clustering Algorithm

Algorithm Framework and Process

In this paper, the text clustering is divided into three stages: text preprocessing, text feature extraction, and text feature vector clustering. These stages are parallelized and their schematic diagram is shown as follows:

The master node collects text data needed from database and these data will be divided and delegated to slave nodes, then each result can be aggregated on the master node.

Text Preprocessing

Unlike English, Chinese words can be composed of multiple characters but with no space appearing between words [7,8]. So word segmentation is considered an important first step for Chinese text processing tasks. In this paper, a module named Ansj [9] for Chinese word segmentation is used to process texts and some meaningless words will be removed such as stop words, auxiliary word, conjunction and so on.

Text Feature Extraction

This paper proposes a parallel TF-IDF (Term Frequency-Inverse Document Frequency) [10] statistical approach to extract feature vector from preprocessed texts. TF represents frequency of each term occurs in each text. For word \( w_i \) in text \( d_j \), its TF is calculated as:

\[
tf_{i,j} = \frac{n_{i,j}}{\sum_{k} n_{k,j}}
\]  

Where \( n_{i,j} \) represents the number of times that word \( w_i \) occurs in the \( d_j \), \( \sum_{k} n_{k,j} \) represents the number of words in text \( d_j \).

IDF is a measure of how much information the word provides, that is, whether the term is common or rare across all texts. For word \( w_i \), its IDF is calculated as:

\[
idf_i = \log \left( \frac{|D|}{|\{d_j | w_i \in d_j\}|} \right)
\]

where \( |D| \) is the number of all documents and \( |\{d_j | w_i \in d_j\}| \) is the number of documents containing the term \( w_i \).
\[ idf_i = \log \frac{N}{df_i} \]  

(2)

Where \( N \) represents the total number of texts, \( df_i \) represents the number of text containing the word \( w_j \).

For word \( w_i \) in text \( d_j \), its TF-IDF value is calculated as:

\[ tfidf_{i,j} = \lambda \cdot tf_{i,j} \cdot idf_i \]  

(3)

Where \( \lambda \) represent the weight value of world. Words in title are usually more important [11], so the weight for title should be increased as follows:

\[
\lambda = \begin{cases} 
2 & \text{if word in title} \\
1 & \text{else}
\end{cases} 
\]  

(4)

For text \( d_j \), its text feature vector is represented as:

\[ \overrightarrow{D_j} = \{tfidf_{1,j}, tfidf_{2,j}, ..., tfidf_{n,j}\} \]  

(5)

The basic steps of parallel feature extraction are shown as follows:

**Algorithm 1**: TF-IDF feature extraction

**Input**: Part of texts for each slave node after preprocessing. The number of features \( L \) that represents the length of feature vector (should be greater than the number of all words in texts).

**Output**: feature vectors.

1. On each slave node, calculate the term frequency for each text and the result is stored by a hash table where the index of word as the key and term frequency as the value. The index of word is calculated as follows: \( index = \text{hashfunc}(\text{word}) \mod L \). Transfor the hash table to sparse vector [12] \( tf_j \) that represents the TF vector of text \( d_j \).

2. On each slave node, calculate the document frequency vector \( df_s \) for slave node \( s \) as follows:

   \[
   \text{for } tf_j \text{ in this slave node } s:\n   \text{for index in this } tf_j:\n   \quad df_s(index)++=1
   \]

3. On the master node, aggregate all document frequency vectors from all slave nodes and accumulate corresponding value to get the IDF vector \( idf \). Send a broadcast variable [2] including \( idf \) to all slave nodes.

4. On each slave node, calculate the TF-IDF feature vector for each text via equation (3).

**Text Feature Vector Clustering**

In this paper, Cosine similarity is used to measure the similarity between two texts. For text \( d_i \) and \( d_j \), their similarity is calculated as:

\[ \text{sim}(d_i, d_j) = \frac{\overrightarrow{D_i} \cdot \overrightarrow{D_j}}{\|\overrightarrow{D_i}\| \|\overrightarrow{D_j}\|} \]  

(5)

If the similarity is greater than the threshold, two texts are considered as a clustering. The basic steps of parallel Single-Pass clustering algorithm are shown as follows:

**Algorithm 2**: parallel Single-Pass clustering
Input: TF-IDF feature vectors on each slave node calculated by algorithm 1. Similarity threshold $T$.
Output: clusters.
1. On the master node, aggregate all TF-IDF feature vectors for each slave node to get vectors $V$. Send a broadcast variable including $V$ to each slave node.
2. On each slave node, for each TF-IDF feature vector in this node, calculate the similarity between this vector and part of vectors in $V$ (the $n$th vector should be compared with the first $n$ vector in $V$) via equation (5). Then get the largest similarity $\max_{i,j}$ which represents the similarity between text $d_i$ and $d_j$. If $\max_{i,j}$ is greater than $T$, text $d_i$ and $d_j$ belong to the same cluster.
3. Get all clusters relying on the transitive relation of similarity.

Experiments and Results Analysis

Environment and Experimental Data Selection
Distributed web crawler is built with 5 server (8 core CPU, 16GB RAM, 300GB Hard disk). And Spark cluster for text clustering is built with 5 server (40 core CPU, 64GB RAM, 300GB Hard disk).
Depending on distributed web crawler, the system crawled 18000 texts on November 6, 2016. From above texts, 10000 texts are chosen to verify the efficiency and 1000 texts of 10 topics are chosen to verify the accuracy. Similarity threshold is set to 0.45.

Evaluation Measure
In parallel computing, speed-up ratio is used to indicate how fast a parallel algorithm is compared to the corresponding serial algorithm [13,14]. It is calculated as:

$$S_p = \frac{T_1}{T_p}$$

where $p$ represents the number of processors, $T_p$ represents execution time with $p$ processors.

Experimental Results and Analysis
In order to analyze the efficiency of parallel Single-Pass clustering algorithm based the Spark cluster computing system, this experiment adopt the traditional stand-alone Single-Pass algorithm and parallel Single-Pass clustering algorithm computed in four kinds of cluster mode with different number of distributed node. The clustering result processing the 10000 texts of 10 topics is shown as follows:

<table>
<thead>
<tr>
<th>Nodes</th>
<th>1G</th>
<th>2G</th>
<th>3G</th>
<th>Average [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>stand-alone</td>
<td>85.832</td>
<td>82.302</td>
<td>80.22</td>
<td>82.784</td>
</tr>
</tbody>
</table>

From table 1, we can know that the memory allowing Spark executor to use just has a little effect on efficiency of this algorithm and the running time is in a fixed range when one cluster mode is decided. So we use the average of running time to get the speed-up ratio diagram in different cluster mode via equation (6).
From figure 3, we can know that this parallel algorithm improved the clustering efficiency significantly, but as the number of distributed nodes increasing, the rate of growth in speed-up ratio became slower and slower. It indicates that we should choose a proper number of distributed nodes for the data of certain amount to maximize the use of computing resources.

![Figure 3. Speed-up ratio under different degree of parallelism.](image)

The results of accuracy in different cluster mode are the same, so this experiment chose the mode with four nodes to compare with the traditional stand-alone Single-Pass algorithm. The clustering result calculated by traditional Single-Pass algorithm and the algorithm in this paper processing the 1000 texts of 10 topics is shown in Table 2.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Count</th>
<th>Count (traditional)</th>
<th>Count (this paper)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contradictions between SF Express and Cainiao</td>
<td>100</td>
<td>96</td>
<td>100</td>
</tr>
<tr>
<td>Panama established diplomatic relations with China</td>
<td>100</td>
<td>68</td>
<td>69</td>
</tr>
<tr>
<td>The Belt and Road strategy</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Domestic large passenger aircraft c919 first flight</td>
<td>100</td>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td>Korean Peninsula issue</td>
<td>100</td>
<td>76</td>
<td>90</td>
</tr>
<tr>
<td>United Airlines violence</td>
<td>100</td>
<td>70</td>
<td>85</td>
</tr>
<tr>
<td>Xiongan New Area plan</td>
<td>100</td>
<td>86</td>
<td>88</td>
</tr>
<tr>
<td>Xi Jinping's visit to Finland</td>
<td>100</td>
<td>76</td>
<td>80</td>
</tr>
<tr>
<td>Bicycle-sharing</td>
<td>100</td>
<td>83</td>
<td>90</td>
</tr>
<tr>
<td>Bak Geunhye impeachment case</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

From table 2, we can see that for the clustering accuracy of each topic the proposed algorithm had a better performance.
Conclusions
This paper designed and implemented the student-oriented Internet public opinion system. The system uses distributed web crawler to collect Internet information and employs parallel Single-Pass algorithm based Spark to detect hot topics efficiently when dealing with massive or high-dimensional data. The experiments validated that compared with the traditional Single-Pass algorithm, ensuring the accuracy, the proposed algorithm improved the clustering efficiency at the same time. And it provided a guarantee for real-time monitoring of public opinion.

References
[9] Information on https://github.com/NLPchina/ansj_seg