CS5296 - Cloud Computing

Semester B 2023-2024

Group Project Software Artifact

Submitted on *30-04-2024*

Weibo Hot Topic Timeline Based on AWS Platform

Group 9

58208460	Yixin LI
58276105	Tianyi LIANG
58097428	Peilin LIU
58336862	Guanchen LYU
58277901	Zihan XIAO

Project Overview:
In today's fast-paced world, social media platforms like Weibo have become central hubs for news dissemination and opinion sharing. However, the rapid influx of information leads to significant challenges, notably information overload. During major news events, user-generated content surges, creating a dense and complex flow of information that can be challenging to navigate and understand comprehensively.
This project addresses these challenges by leveraging advanced technologies to crawl, process, and analyze data from the Weibo platform to construct insightful event timelines. Our primary objective is to capture daily hot topics, analyze the content, and provide users with a clear, chronological view of events as they unfold.
Goals:
1. Data Crawling: Develop a custom web crawler to collect daily trending topics from Weibo, including posts and associated metadata.
2. Data Processing and Storage: Utilize robust big data technologies like Hadoop or Spark to manage the volume and complexity of the data.
3. Text Analysis: Apply advanced text analysis techniques to understand user behavior and topic evolution on Weibo.
4. Machine Learning: Train machine learning models on distributed systems to enhance the accuracy and efficiency of the timeline generation process.
5. Cloud Integration: Implement cloud-based solutions to improve the scalability and responsiveness of our system, ensuring that users can access and interact with the timeline seamlessly.
6. By adopting these methodologies, we aim to mitigate the issue of information overload on Weibo, making it easier for users to grasp the significance of news events without getting lost in the vast sea of information. Our system will provide a valuable tool for both casual users and professionals who rely on timely and accurate information aggregation to make informed decisions.

1. Environment Setup
To ensure the successful deployment and operation of our Hot Topic Timeline Generator, specific environment configurations are required. Below is a detailed guide on setting up the environment using Amazon Web Services (AWS), specifically EC2 and S3 services.

1.1. AWS EC2 Configuration

Instance Type: m5.large
This instance type provides a balance of compute, memory, and networking resources, making it ideal for the medium-scale processing required by our project.

Operating System: Ubuntu
AMI Name: ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-20240301
This AMI provides Ubuntu 22.04 LTS, offering a stable and secure environment for our application.

1.2. AWS S3 Configuration
Usage: Storing and retrieving data such as vocabulary lists, weight matrices, and inverted

index tables.
AWS S3 is utilized for its scalability, security, and high availability, which are crucial for handling large volumes of unstructured data generated and processed by our application.

1.3. Local and Remote Development Environment Setup

Ensure that your local development environment matches the production settings as closely as possible. This includes installing necessary software and libraries that are compatible with the AWS EC2 instance.
Required Software:

1. Python 3.8 or higher
2. Apache Hadoop
3. Apache Spark
4. MXNet
5. TensorFlow
6. Additional Python libraries as specified in requirements.txt

2. Installation Guide

This guide provides detailed steps for setting up and configuring the AWS EC2 instance and necessary software for the Hot Topic Timeline Generator project.

2.1. Connecting EC2 with VSCode Remote SSH

To manage development directly on the AWS EC2 instance, use the VSCode Remote - SSH extension:

SSH Configuration:

Save your EC2 SSH key (e.g., your-ec2-key.pem) in a safe directory on your local machine:

C:/Users/username/.ssh/

Configure SSH to connect to your EC2 instance by editing the SSH config file:

C:/Users/username/.ssh/config

Add the following configuration:

```
Host aws-ec2
   HostName <your-ec2-ip-address>
   User ubuntu
   IdentityFile ~/.ssh/your-ec2-key.pem
```

2.2. Basic Environment Setup on Remote Server

Once connected through SSH, prepare the server environment:

```
sudo apt-get update
sudo apt-get install sysbench
sudo apt install python3-pip
```

2.3. Setting up Project Dependencies
Navigate to the project directory and install the required Python dependencies:

```
cd TopicTimeline
sudo apt install python3-pip
pip3 install -r requirements.txt
```

2.4. Handling Missing Chinese Fonts
If running the code results in errors due to missing Chinese fonts, you need to install the font and clear the matplotlib cache:

```
sudo cp /home/ubuntu/TopicTimeline/SimHei.ttf /usr/share/fonts/
rm ~/.cache/matplotlib/fontlist-v330.json
rm -rf /home/featurize/.cache/matplotlib
```

2.5. Configuring EC2 and S3 Connectivity
For seamless integration between the EC2 instance and AWS S3, follow these steps:

IAM and Role Assignment:
1. Create an IAM role with the AmazonS3FullAccess policy.
2. Assign this IAM role to your EC2 instance to facilitate easy access to S3 buckets.

Using Boto3 for AWS Services
Install and configure Boto3, a Python library that allows you to directly interact with AWS services:

```
pip3 install boto3
```

This setup will ensure that your EC2 instance can efficiently interact with S3, utilizing it for storing and retrieving large datasets necessary for the project.

2.6. Configuring the HADOOP Environment
Make sure Java is installed on your EC2 instance.
Download the HADOOP package and unzip it:

```
wget " https://mirrors.sonic.net/apache/hadoop/common/hadoop-3.4.0/hadoop-
3.4.0.tar.gz"
sudo tar -zxvf hadoop-3.4.0.tar.gz -C /opt
```

Add these commands to the end of the ~/.bashrc file:

```
export HADOOP_HOME=/yourpath/hadoop-3.4.0
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

Reload the .bashrc configuration with the following command:

```
source ~/.bashrc
```

Configure core-site.xml, hdfs-site.xml, mapred-site.xml, yarn-site.xml files:
Most components of the Hadoop ecosystem are configured using .xml files. They use a uniform format: that is, configuration items are filled in between <configuration> and </configuration> in each configuration file, and each configuration item is presented in the following way:

```
<property>
<name>Configuration item name</name>
<value>Configuration value</value>
</property>
```

We need to configure the corresponding configuration item in the corresponding file
core-site.xml:

fs.defaultFS	hdfs://localhost/

hdfs-site.xml:

dfs.replication	1
dfs.name.dir	/home/hadoop/hdfs/namenode
dfs.data.dir	/home/hadoop/hdfs/datanode

mapred-site.xml:

mapreduce.framework.name	yarn

yarn-site.xml:

yarn.nodemanager.aux-services	mapreduce_shuffle
yarn.resourcemanager.hostname	localhost

Initialize Namenode node data:

```
hdfs namenode -format
```

Starting Hadoop services:

```
start-dfs.sh
```

To verify that the Hadoop-related services were started successfully, we use the jps command provided in the JDK:

```
jps
```

The result should be like this:

```
1585 DataNode
27186 Jps
26037 NodeManager
1447 NameNode
1772ondaryNameNode
```

2.7. Configuring the SPARK Environment
Download the HADOOP package and unzip it

```
wget "https://archive.apache.org/dist/spark/spark-3.5.1/spark-3.5.1-bin-hadoop3-
scala2.13.tgz"
sudo tar xvf spark-3.2.0-bin- spark-3.5.1-bin-hadoop3-scala2.13.tgz
```

Add these commands to the end of the ~/.bashrc file:

```
export SPARK_HOME=/yourpath/spark
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
```

Reload the .bashrc configuration with the following command:

```
source ~/.bashrc
```

3. Code Execution

To execute the code for the Hot Topic Timeline Generator project, follow the steps below according to the provided file structure and the role of each script in the project.

Project File Structure Overview:
scrapy.py: Python script to scrape information related to trending topics on Weibo using the requests library and BeautifulSoup library, then save the results to a CSV file.
pyspark_code.py: Python script for processing text data, which includes reading, tokenizing, feature extraction (TF-IDF), data merging, etc. using pyspark.

add_new_mind.ipynb: Jupyter notebook used to train the sentiment analysis model using TensorFlow.
mxnet_code.py: Python script for training the sentiment analysis model using MXNet.
process.py: Server-side script for uploading vector matrices.
query.py: Client-side script for querying hot topics within a specified time period; results are stored in picture and result_excel directories.
README.md: Contains the project description and instructions.
requirements.txt: Lists all the Python dependencies required for the project.
Other scripts and resources are part of the project's supporting files.

3.1. Data Crawling
Running scrapy.py to get Weibo hot topics data:

python scrapy.py

Please note that due to Weibo's own anti-crawler mechanism and network connection problems, the crawling process may be stalled or failed, please try to run the file after using a VPN.

3.2. Training Vector Matrices
Make sure your python environment has downloaded the pyspark library.
Run pyspark_code.py to get Vector Matrices:

python spark_code.py

3.3. Training the Sentiment Analysis Model with TensorFlow
To train the sentiment analysis model using TensorFlow:

jupyter notebook add_new_mind.ipynb

Follow the instructions in the notebook to train and evaluate the model.

3.4. Training with MXNet (Optional)
If you wish to experiment with the MXNet model, run:

python mxnet_code.py

However, for this project, the TensorFlow model provided better results and will be used for further processing.

3.5. Uploading Vector Matrices
Execute the process.py script on the server to upload vector matrices:

python process.py

Ensure the server environment is correctly configured as described in the installation guide.

3.6. Querying Hot Topics and Generating Results
Run the query.py script to query hot topics and generate visualizations:

```
python query.py
```

The outputs will be saved in the picture directory as visualizations and result_excel as spreadsheets.

Remember to install all required Python dependencies listed in requirements.txt before running the scripts. Use the following command to install dependencies:

```
bash pip install -r requirements.txt
```

Ensure that the appropriate fonts are installed if you encounter issues with missing fonts during visualization generation, as previously described in the installation guide.