Institute for Data Engineering
Module Big Data – Lab Exercises

Project Topic A – Federated Sentiment Analysis

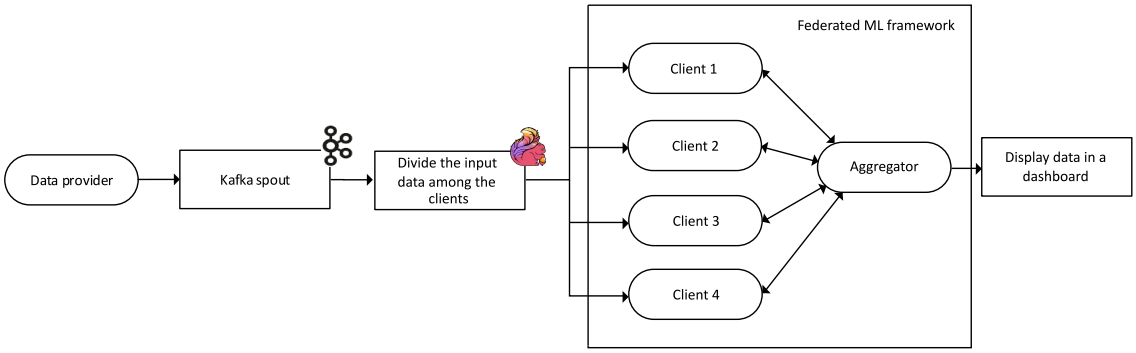

Figure 1: Federated Sentiment Analysis

Introduction

Sentiment analysis is a method used in natural language processing (NLP) to identify, extract, quantify and study affective states and subjective information [3]. It is often applied to understand opinions, emotions, and attitudes expressed in written text. In this project, our goal is to identify if the sentiment in certain pieces of text is positive or negative towards a certain topic. For instance, if we define the keywords "Trump" and "Biden", the implemented software should provide us with a statement "98% positive" (or similar) for each of the keywords, based on a repository of text fragments. For this, a cloud-based solution needs to be implemented, and a *federated learning* approach needs to be followed.

In this project, the observed text fragments are Reddit comments [1]. The according data you need for the sentiment analysis is provided via the the website: 'The Pushshift Reddit Dataset'. It contains two '.zst' files, one containing the submissions (RS_2019-04.zst) and the other for the comments (RC_2019-04.zst), which when uncompressed amounts to approx. 205 GBs in total.Since the complete dataset is too big for this project, you will only use a part of the data from the reddit comments, i.e., RC_2019-04.zst. This is explained further in section stage 1 below.

For the sentiment analysis, machine learning is applied. However, instead of going for a centralized machine learning approach, federated learning [2] is used. The basic idea of federated learning is to divide the training of a model by giving clients each a particular data package to generate a *local model* for each client. This local model is then sent to an aggregator, who generates a global model from it. This global model is then sent back to the clients for inference purposes.

Within the work at hand, we make use of a stream processing approach for this. The input in terms of data streams is provided to different "client operators" which learn each a local model based on their respective input data streams. The local model is forwarded to an aggregator operator, which generates a global model. This global model can then be used to compute the sentiment. Naturally, over time, further Reddit comments will have to be handled by the client operators, and the global model needs to be updated. This allows to show how sentiments change over time. For this, it is necessary to provide the client operators with the global model, i.e., the global model becomes the foundation for training in the next iteration.

The preparatory task for this project is to populate an Apache Kafka instance with Reddit

comments. Your task is to issue these comments according to their timestamps, and to submit the replayed data stream to Apache Kafka. Notably, training is done in a distributed way, i.e., each client operator actually gets only part of the overall data package for their local training. How you divide the data is up to you, but you should not do it based on timestamps. Instead, you could use data from different Reddit communities, comments provided in different languages, etc. Apache Kafka acts then as a data source for Apache Flink, which processes the comments to train the local models and to aggregate the global model.

Outcomes

The expected outcomes of this project are two-fold: (1) the actual project solution, (2) an intermediary and a final presentation of your results.

Project Solution

The project has to be hosted on a Git repository in TUHH's GitLab instance[1] – you will get instructions about the repository setup at the lab kickoff meeting. Every member of your team has to use its own, separate Git account. *We will check who has contributed to the source code*, so please make sure you use your own account when submitting code to the repository. Furthermore, it is required to provide an easy-to-follow README that details how to deploy, start and test the solution. The best practice is to provide a README that describes "Plug-and-Play" instructions on how to use your solution.

For the submission (see below), you also have to create one or more Docker images, which contain(s) your complete implemented solution, i.e., including all dependencies.

Presentations

There are two presentations. The first one is during the consultation hours, and describes your status at that particular point of time. This presentation will not be graded, i.e., it is primarily a way to check your progress and to show to other groups what you are working on and how you want to solve the problem.

The second presentation is during the final meetings and contains all your results. The actual dates are announced in Stud.IP.

Every member of your team is required to present in either the first *or* the second presentation. Each presentation needs to consist of a slides part and a demo of your implementation. Think carefully about how you are going to demonstrate your implementation, as this will be part of the grading. You have 10 minutes (strict) of time for your presentation in the consultation hour, and 15 minutes (also strict) of time for your presentation at the final meeting. At the first presentation, you can go for a slide-based presentation, a live demo, or a combination of both, depending on your progress until that point of time.

The past has shown that providing a nice use case story usually helps to present the project outcomes. While providing such a use case story is not an absolute must, it will surely help the audience to understand your work better.

Grading

A maximum of 60 points are awarded in total for the project. Of this, 70% are awarded for the implementation (taking into account both quality and creativity of the solution as well as code quality and documentation), and 30% are awarded for the final presentation (taking into account content, quality of slides, presentation skills, and discussion).

A strict policy is applied regarding plagiarism. Plagiarism in the source code will lead to 0 points for the particular student who has implemented this part of the code. If more than one group

[1] `https://collaborating.tuhh.de/`

member plagiarizes, this may lead to further penalties, i.e., 0 points for the implementation of the whole group.

Deadline

The hard deadline for the project is **July 9th, 2024**, 23:59. Please submit the presentations via Stud.IP. The Docker images need to be uploaded to DockerHub and made available to the lecturer team. For this, please write a mail on how to access the containers to nisal.hemadasa@tuhh.de. The deadline for this is also July 9th, 2024, 23:59. Late submissions will not be accepted.

Test Cloud Infrastructure

This year, the Google Cloud Platform is supporting the Big Data lecture with an Education Grant, which you can use for Virtual Machines (VMs) and other cloud resources. The computational resources can be used for free, however, you need to own a Google account to use them. To access the resources, please visit the following URL (please note that this is a masking URL, i.e., you need to click it, simply copy/pasting it will *not* help): http://google.force.com/. The full URL is also provided in Stud.IP.

Please note that you need to request the resources until August 2nd, 2024; the coupons are then valid until April 2nd, 2025. Afterwards, they will become void and you will not be able to access the budget.

The e-mail address you provide in this form does *not* have to be your Google account, but it *must* be an address within the TU Hamburg domain, i.e., `@tuhh.de`. By providing such an address, you can claim a coupon code, which you can then use to redeem a $50.00 voucher for the Google Cloud Platform. Detailed instructions are provided at this URL and subsequent e-mails you will receive from Google.

You can redeem one coupon code per e-mail address at a time. The voucher of $50.00 is in most cases sufficient to conduct the implementation tasks of the lab exercises. However, if this is not the case, please send a mail to nisal.hemadasa@tuhh.de. Please also take into account that there are the free tier resources of the Google Cloud Platform, which are usually sufficient to test and run your solutions.

Note that exceeding this budget will lead to immediate shutdown of the project by Google, and we have no means of influencing this, or providing extensions. We therefore recommend you to monitor your resource usage, in order to avoid unnecessary spending, for instance, by VMs left running.

You are allowed to use the Google Cloud Resources only for the purpose of this module. Both Google and TU Hamburg are monitoring the use of the resources, and any misuse (hosting of illegal data, cryptocurrency mining, etc.) will be punished. In case of any questions regarding the organization of the Google Cloud Platform, please send an e-mail to nisal.hemadasa@tuhh.de.

Please take care that you do not publish any credentials or (ssh) keys on the Internet. For your implementation, please use dedicated credential files and add these credential files and the keys to the .gitignore file or put them on a system path outside of your Git repository. If you accidentally leak any credentials or ssh keys, inform us immediately. Google is pretty good at identifying leaked credentials, and will then shut down your account at least temporarily. If this happens just before the deadline, there is a good chance that your account will not be released again in due time.

In case of noncompliance, the person(s) responsible will fail the lab!

Stage 1

Your task in Stage 1 is to setup and configure Apache Kafka as well as Apache Flink, whereas it is sufficient to run Apache Flink only in local mode for Stage 1 Furthermore, you have to implement the data provider to fill Apache Kafka with data, and run a first basic federated learning solution.

Please be aware that the Pushshift Reddit Dataset comprises a considerable amount of redundant data, that might not be of use for sentiment analysis model training. As a hint, you may filter the values from the following keys (the underlying data is JSON formatted). To downsize the dataset even further, use data between "created_utc":1554076800 - 1555472130.

- id: Each comment on Reddit is given a unique ID that can be used to refer to it specifically.
- author: This field represents the username of the person who made the comment.
- created_utc: This is a timestamp indicating when the comment was posted, measured in seconds since the Unix epoch.
- body: This is the text content of the comment.
- score: This field represents the net score of the comment, which is the number of upvotes minus the number of downvotes the comment has received.
- subreddit: This indicates the subreddit where the comment was posted.
- controversiality : This is a binary indicator (0 or 1) used to show whether a comment is controversial, where controversial comments are those with a nearly equal number of upvotes and downvotes, indicating divided opinion. A value of 0 indicates that the comment is not considered controversial.

Also, note that the information such as emojis are not lost when preprocessing the dataset.

Please note: Stage 1 is what you should have achieved roundabout half-way through the semester. This is a recommendation, not a must. But it will help to avoid too much crunch time at the end of the semester.

Tasks:

1. Setup and configure Apache Kafka.
2. Implement the data provider, which represents a producer for Apache Kafka. This data provider retrieves the information and submits it to Apache Kafka. The data needs to be replayed in the same order as it was recorded, i.e., all comments with the same timestamp need to be submitted at the same time. It is advisable to make the submission speed configurable to apply different submission speeds during testing and actual performance tests of the system.
3. Setup and configure Apache Flink and implement a first simple federated learning solution.
   - You may use Flink's *Apache Kafka Connector* to feeds real-time data into your Flink topology.
   - Implement the required operators for the topology. This means that there should be four client operators and one aggregator operator. As written above, divide the input data among the four clients. You may use a structured approach for this or just distribute the data arbitrarily.
   - For now, it is perfectly fine if the federated learning is done only once, i.e., there are no iterations of the local and global models. However, the sentiment analysis should already work, i.e., there should be an operator which waits for input in order to show the sentiment towards different keywords. You are not allowed to implement an already existing sentiment analysis tool into your solution, i.e., you should really train the models by yourselves. For this, select your own machine learning framework, e.g., TensorFlow Federated.

- Implement a preprocessing approach for the comments. The traditional way to implement stemming and stop word removal is to use pre-defined dictionaries (which are, of course, language specific), which you can find on the Internet. However, keep in mind that Reddit comments work a little differently than more formal texts. For instance, ☺, ☹, etc. should probably not be removed from a tweet, since they could provide very helpful information during the sentiment analysis.

Stage 2

In Stage 2, your solution needs to be deployed in the cloud. Also, an easy-to-use dashboard needs to be implemented, allowing the user to state two different keywords in order to show the sentiments for both of them, and how the sentiments for these keywords evolve over time. For the latter, it is necessary to update the local and global models. You may decide on your own if this is done after a particular timeframe, after a certain number of data items (here: Reddit comments) is processed, or if something else has happened.

Tasks:

1. Implement a basic Web-based GUI for the sentiment analysis. It does not have to be very pretty, but should provide the information described above.
2. Make sure that your solution is running in the cloud in a distributed fashion, i.e., different operators run in single Docker containers.
3. Allow for iterations of the local and global models, i.e., training needs to be done several times, based on your decision when to retrain.
4. Think about further analysis you could conduct. E.g., based on how you have divided the data, you may show us the sentiments in comments in different languages.

References

[1] Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. The pushshift reddit dataset. *Proceedings of the International AAAI Conference on Web and Social Media*, 14:830–839, 2020.

[2] Dinh C. Nguyen, Ming Ding, Pubudu N. Pathirana, Aruna Seneviratne, Jun Li, and H. Vincent Poor. Federated Learning for Internet of Things: A Comprehensive Survey. *IEEE Commun. Surv. Tutorials*, 23(3):1622–1658, 2021.

[3] Mayur Wankhade, Annavarapu Chandra Sekhara Rao, and Chaitanya Kulkarni. A survey on sentiment analysis methods, applications, and challenges. *Artif. Intell. Rev.*, 55(7):5731–5780, 2022.