THE UNIVERSITY OF SYDNEY

School of Computer Science
Dr. Ying Zhou

COMP5349: Cloud Computing Sem. 1/2022

Assignment 2: Data Preproceessing and Performance Tuning with Spark

Individual Work: 20% 10.05.2022

1 Introduction

This assignment tests your ability to handle input data with complex structures using Spark. It also tests your understanding of Spark execution and your ability to tune your implementation and execution environment to improve performance.

The data set you will work on is from the same Atticus Project involving Contract Understanding Atticus Dataset (CUAD). The contract understanding problem is modelled as an extractive Question Answering task. Many natural language processing (NLP) tasks involve a data pre-processing step to convert the raw data to proper model inputs. If the pre-processing step is executed sequentially on a CPU, it may need a few hours to process all training data. This is the case for the CUAD data set.

You are asked to develop a Spark program that can execute on multiple nodes to pre-process the CUAD data in parallel. You also deep to provide a performance report to analyse your program's resource usage and scalability.

2 Input Data Set Description

The CUAD data set presented in typical question answering format `data.zip` can be downloaded from the project repository. The `data.zip` contains the following three files:

- `CUADv1.json`: the complete data set
- `train_separate_questions.json`: the training set
- `test.json`: the test set

All files are of the same SQuAD-style JSON format. The complete data set contains data of all 510 contracts. The training set contains data of 408 contracts. The test set contains data of 102 contracts.

Each contract is represented as a nested JSON object of the following format:

```
{"title":"contract title" ,
 "paragraphs": [
    {
        "context": "the full contract as a string",
        "qas": [
            {
                "id": "document name and category label",
                "question": "question text",
                "is_impossible": True/False,
                "answers": [
                    {
                        "answer_start": answer start position ,
                        "text": "the actual answer text"
                    },
                ...
                ]
            },
        ...
        ]
    }
  ]
}
```

There are two fields at the top level: “title” and “paragraphs”. The “title” field stores the title of a contract. The “paragraphs” field stores the main data and it is of array type; each element in the array represents a paragraph in the document. The CUAD dataset treats the entire contract as a single paragraph. This array always contains a single *paragraph* object.

Each paragraph object has two fields: “context” and “qas”. The “context” field stores the entire contract text as a string. The “qas” field is of array type and stores the question and answer pairs related with this contract. The complete CUAD data set contains 13,000+ clauses belonging to 41 categories. Each category is considered as a question. The “qas” field of any contract contains 41 question answer pairs; each represents a category and is stored as a nested JSON object.

The question answer pair object has four fields: “id”, “is_impossible”, “question” and “answers”. The “id” field stores a unique id of the question. Its value is a string concatenating the document file name and the category label. The “question” field stores the question text. For instance, the “question” field for category “Document Name” has the following text: “Highlight the parts (if any) of this contract related to "Document Name" that should be reviewed by a lawyer. Details: The name of the contract”. The “is_impossible” field is of

Boolean type. A `True` value indicates there is no answer for this question in this contract. The "answers" field is of array type and stores potential answers. If the "is_impossible" field is `True`, the "answers" field contains an empty array. Otherwise, it contains one or more answer objects; each represents a valid answer of that question (an annotated clause belonging to that category).

Each answer object has two fields: "answer_start" and "text". The "answer_start" field stores the starting position of the answer in the contract (the index of the answer's first character). The "text" field stores the actual answer text.

The actual JSON file also contains the version information. The root object of the file has two fields: "version" and "data". All contract data as described above are stored as an array in the "data" file.

3 Workload Description

A typical training sample of a question answering model is a four element tuple in the following format: (source, question, answer start location, answer end location). The start and end location values are set to 0 for a question with no answer in the context. If a question has two answers in the same context, two samples will be created with different answer start and end locations.

The CUAD training data set has 408 contracts, each with 41 questions and their corresponding answers or no answer indicator. If each question has at most one answer and the entire contract is set as source, the data set would theoretically contain $41 \times 408 = 16728$ training samples.

However, most NLP models only accept input that has a specified maximum length, which in turn requires maximum length to be set on the source text and the question text. A contract is much longer than the maximum length of a source text any model could accept. A typical pre-processing step involves segmenting long input into multiple sequences of fixed size and using each sequence as the source text in a sample. If a contract is segmented into 100 sequences, and all questions have at most one answer in this contract, $100 \times 41 = 4100$ samples will be generated for this contract.

In this assignment, you are asked to segment each contract into overlapping sequences of 512 characters. This should be done using a sliding window of**512** bytes(characters) and a stride of **256** bytes(characters). A stride is the distance the window should move to generate the next sequence. This way, a contract containing 1036 characters would be segmented into 5 sequences. The first one contains the first 512 characters. The 2nd sequence contains characters between location 256 (inclusive) and 768 (exclusive). The third and forth sequences start from locations 512 and 768 respectively; each of them contains 512 characters. The last sequence contains the rest of the characters from location 1024.

Not all sequences contain an answer. In fact, for any particular question, only one or

a few sequences may contain the answer(s). Samples with a source sequence containing some part of an answer to a question are called the *positive samples* of that question. Samples with source sequences that do not contain any part of an answer to a question are called the *negative samples* of that question. In the CUAD data set, the labelled clauses make up about 10% of each contract on average[1]. The process of segmenting the contract text into sequences to create training samples would generate many more negative samples than positive samples.

The next pre-processing step involves balancing the negative and positive samples. In other words, we need to keep all positive samples but only a small portion of the negative samples as training data.

We consider two types of negative samples:

- *Impossible negative*. A contract might contain zero occurrences of a particular category. This is indicated by the "is_impossible" field. In this case, all samples generated from this contract for this question are negative.

- *Possible negative*. For a contract where "is_impossible" is set to `False` to a question, only a small number of sequences contain all or part of the answers to that question. Only a small portion of samples generated for this question are positive. All other samples are negative.

We want to include both types of negative samples and ensure a relatively balanced number of positive and negative samples for each question in each document. We use the following simple heuristics to decide the number of negative samples to keep for each question in each contract:

- For an impossible question in a contract, the number of *impossible negative* samples to keep equals the average number of positive samples of that question in other contracts.

- For each contract, the number of *possible negative* samples to keep for each question equals the number of positive samples of that question in this contract.

Note this will result in a slightly higher number of negative samples in the data set due to the inclusion of impossible negatives.

We also want to ensure maximum coverage of the contract text in the training samples. We expect that each negative sample of the same contract contains a unique sequence and that this sequence should not have appeared in any positive sample of the contract.

4 Coding Requirements

You are asked to write a Spark program to generate samples as described in section 3 for contracts in the CUAD data set. You can use either Spark RDD or SQL API. Your program

should take either JSON file in the CUDA data set as input and generate samples based on the JSON file. The generated samples should be written in a JSON file using the following format:

```
[
    {
        source: "... ",
        question: "... ",
        answer_start: s,
        answer_end: e
    },
    ...
]
```

The program should be written as a single Python script that can be submitted to a Spark cluster. You should also provide a submit script to allow markers run your script in a cluster setting.

You are encouraged to print out useful intermediate results to help the marking process.

5 Report

You are also required to submit a report with the following sections:

- Introduction. A brief introductory section.
- Design. In this section, briefly describe the data flow of your implementation. You should include descriptions on key DataFrames or RDDs created and the operations used.
- Execution. In this section, describe the execution plan of the main job(s) that produces the samples as described in section 3. You may use screenshots from Spark history server output.
- Performance Observation and Tuning. You are asked to run your program in a multi-node cluster. The cluster may have between 3 - 9 nodes. You are asked to investigate the resource consumption, particularly your program's memory consumption and use various Spark configuration options to achieve optimal performance[2]. The resource consumption and the configuration options should be documented in this section.
- Scalability. You are asked to run your application on a small data set (`test.json`) and a large data set (`train_separate_questions.json` or `CUADv1.json`) in the same cluster. You may adjust the configuration based on data set size. Compare

the performances in terms of execution times and investigate the scalability of your program.

- Summary. An optional section to summarize the report

6 Deliverables

Two submission inboxes will be open in the COMP5349 Canvas Assignments page. One of them is for a zip file containing your program in a single Python script and a submit script. The other one is for your report. Your report must be in pdf format. You must submit your files in the correct formats to the appropriate submission inboxes. The due date is week 13 Friday 27/05/2022 23:59 Sydney time.

If you submit (or resubmit) your work after the deadline, a late penalty will be applied. This penalty will depend on the time of your latest submission.

Please double check that the code you submitted is the correct version and can run on an EMR cluster without any additional dependency packages. The markers will run your code as part of the marking process. Only the submissions that can execute and produce results will be marked. Your marker may contact you for an optional demo if they cannot run your script. If you can demonstrate that the error was caused by minor mistakes not related with programming logic, your submission will be marked accordingly. Penalties will be applied in this case. If you miss your demonstration, your mark will be based on the part of your submission that produces some output.

References

[1] Hendrycks, Dan and Burns, Collin and Chen, Anya and Ball, Spencer, Cuad: An expert-annotated nlp dataset for legal contract review, arXiv preprint arXiv:2103.06268 (2021).

[2] Tuning Spark