



MPI Programming Project

20.12.2022

Course : CMPE300 - ANALYSIS OF ALGORITHMS

Type: Programming Project

Halis BAL - 2020400363

Emre SİN - 2019400207

Introduction

Our purpose was to develop a program that calculates the data for an n-gram language model using Python. More specifically, a bi-gram language model utilizing MPI framework and doing the calculations in parallel.

Program Interface

The program is executed on the command line, using mpiexec command. The program takes 4 different input arguments from the user.

- -n: Number of processes
- -input_file: Path to the input file to read the sentences
- -merge_method: Method to use while merging the outputs of the workers.
- -test_file: Path to the test file to calculate bigram frequencies

Example command:

```
mpiexec -n 5 python main.py --input_file data/sample_text.txt --merge_method MASTER  
--test_file data/test.txt
```

Program Execution

Input file contains different sentences on each line. The program's functionality is to calculate unigram and bigram counts in each sentence and calculate the frequency of each bigram given in the test file. Firstly the program will print the ranks and task distribution for each worker, then the program will print each bigram's frequency in the console.

Users should use the example command format, then should run the code in the terminal. Frequencies of the given bigrams will be printed automatically and the program will terminate.

Input/Output

There are 2 different text input files.

- 1- Input File: Contains sentences using '<s> text </s>' format on each line.
- 2- Test File: Contains bigram examples on each line.

Sample Input File (.txt)

<s> türk halk müziği ve protest müziğin önemli isimlerinden </s>
<s> güvercinleri de vururların söz ve müziği ise şehrazata ait </s>
<s> toplumsal duyarlılığı ve muhalifliğiyle tanınan selda bağcan daha </s>
<s> meadow rain walker avukatı aracılığıyla verdiği </s>
<s> meadow rain walkerın avukatı jeff milam yatpığı </s>
<s> caddelere uygun olarak tasarlanmamış diye konuştu </s>

Sample Test File (.txt)


pazar günü
pazartesi günü
karar verecek
karar verdi

Sample Output (in the console)

Rank: 1, Num Sentences: 59109
Rank: 2, Num Sentences: 59109
Rank: 3, Num Sentences: 59109
Rank: 4, Num Sentences: 59108
pazar günü: 0.4462962962962963
pazartesi günü: 0.5966101694915255
karar verecek: 0.010940919037199124
karar verdi: 0.13216630196936544
boğaziçi üniversitesi: 0.37272727272727274
bilkent üniversitesi: 0.2222222222222222

Program Structure

Program consists of a single code file. In the first part, it parses the arguments given by the user using sys and getopt libraries. After that, there are 3 helper functions that are used on text splitting and task distributing. Then, we have an if statement to check the merge method. Based on the given merge method, we implemented two different approaches. In both of the approaches, there is another if block to check whether the process is master or



worker. We did this by checking their ranks. If it is 0 then it is the master process. Otherwise, it is the worker process. Firstly, the master process reads the input file and distributes the tasks to each worker process using mpi4py library's send function. Then it starts waiting for the outputs. And each worker gets their tasks using mpi4py library's recv function, prints their rank and number of tasks it has received, then starts to count their unigram and bigram counts.

In the master merging method approach, the master process waits for output from each process in a for loop. After receiving all the outputs, it merges them and calculates the necessary frequencies and prints them.

In the worker merging method approach, the master process only waits for one output from the highest rank process because each worker process sends their outputs to the next process. In each worker process, after they calculate their task results, they get the merged output from the previous process then merge with their own results. Then send them to the next process. At the last process, final data is being created and being sent to the master process. Master process gets the final output and calculates the necessary frequencies and prints them.

Frequency Calculation

The frequency of a bigram is calculated using the unigram and bigram counts in the worker processes. Unigram counts means basically the count of each word in the text. Bigrams consist of 2 words, and their frequency means conditional probability of having a second word after the first word. So we calculated the frequency with this formula:

$$(\text{bigram count}) / (\text{unigram count of first word of the bigram})$$

Data Structures

Task distribution format: {rank_id: tasks[]}

- task_distributor function returns this format and the master process uses this format to send the tasks to workers.

Unigram/Bigram Count Format: {'unigram/bigram text': count}

- Example: {'pazar günü': 16} (Which means bigram 'pazar günü' is found 16 times in the text.
- This format is used in all worker processes while calculating the bigram and unigram counts. They use this format to store their calculation results. And also this format is used to transfer data between processes.

Improvements and Extensions

One area for improvement in the Python code is the use of functions. By making use of functions, the code could be more modular and easier to read. Additionally, the use of classes can help improve the design of the code and make it more efficient. Currently, there are some sections of the code where the same tasks are repeated, which could be refactored using functions or classes. By implementing these best practices, the code can be made shorter and more understandable.

Difficulties Encountered

Because we developed the code together, it was sometimes hard to work collaboratively. Since each part of the code is connected to each other, splitting it brought some difficulties with it. Sometimes we were not able to work concurrently.

Conclusion

With this project, we learned how the MPI framework works. This project enabled us to feel more comfortable with the concepts of parallel programming. It was helpful to work collaboratively on a single project file, this led us to use Github.

Appendices

Source code is submitted separately, it can be found in the zip file.