Analyze letter frequency through Apache Hadoop

Cloud Computing's project

Giovanni Bergami
g.bergami@studenti.unipi.it

Marco Bologna
m.bologna2@studenti.unipi.it

Gabriele Frassi
g.frassi2@studenti.unipi.it

Department of Information Engineering, University of Pisa

A.Y.2023-2024

Introduction

1. **Implementation of a LetterFrequency counter using MapReduce**
   - inMapper combiner
   - combiner
2. **Construction of datasets**
   - Specific languages
   - Different dimensions, from KBs to GBs
3. **Data analysis**
   - Languages differences
   - Number of reducers, inMapper combiner vs combiner
   - Performance comparison (time, memory, distributed vs non distributed approaches)

Pseudocode implementation *external combiner*

Algorithm 1 LetterCount

```
1:  Class LetterCount
2:    Class LetterCountMapper extends Mapper
3:      Variables:
4:        one ← 1
5:        tot_letters ← "total_letters"
6:
7:      Method map(key, value, context):
8:        text ← convert value to lowercase string
9:        for each character c in text:
10:         if c is a letter:
11:           write (tot_letters, one) to context
12:
13:   Class LetterCountReducer extends Reducer
14:     Variables:
15:       result ← 0
16:
17:     Method reduce(key, values, context):
18:       sum ← 0
19:       for each val in values:
20:         sum ← sum + val
21:       Set result to sum
22:       write (key, result) to context
```

Algorithm 2 LetterFrequency

```
1:  Class LetterFrequency
2:
3:    Class LetterFrequencyMapper extends Mapper
4:      Variables:
5:        one ← 1
6:        letter ← empty text
7:
8:      Method map(key, value, context):
9:        text ← convert value to lowercase string
10:       for each character c in text:
11:         if c is a letter:
12:           letter ← strip accents from c
13:           write (letter, one) to context
14:
15:   Class LetterFrequencyCombiner extends Reducer
16:     Variables:
17:       result ← 0
18:
19:     Method reduce(key, values, context):
20:       tot ← 0
21:       for each val in values:
22:         tot ← tot + val
23:       Set result to tot
24:       write (key, result) to context
25:
26:   Class LetterFrequencyReducer extends Reducer
27:     Variables:
28:       totalLetters ← 1.0
29:
30:     Method setup(context):
31:       conf ← get configuration from context
32:       totalLetters ← get double value "totalLetters" from conf, default is 1.0
33:
34:     Method reduce(key, values, context):
35:       tot ← 0.0
36:       for each val in values:
37:         tot ← tot + val
38:       result ← tot / totalLetters
39:       write (key, result) to context
```

Pseudocode implementation *inMapper combiner*

Algorithm 3 InMapperLetterCount

```
1:  Class InMapperLetterCount
2:    Class LetterCountMapper extends Mapper
3:      Variables:
4:        count ← 0
5:        tot_letters ← "total_letters"
6:
7:      Method setup(context):
8:        Initialize count to 0
9:
10:     Method map(key, value, context):
11:       text ← convert value to lowercase string
12:       for each character c in text:
13:         if c is a letter:
14:           Increment count by 1
15:
16:     Method cleanup(context):
17:       Write (tot_letters, count) to context
18:
19:   Class LetterCountReducer extends Reducer
20:     Variables:
21:       result ← 0
22:
23:     Method reduce(key, values, context):
24:       sum ← 0
25:       for each val in values:
26:         sum ← sum + val
27:       Set result to sum
28:       write (key, result) to context
```

→ *setup method*

→ *cleanup method*

Algorithm 4 InMapperLetterFrequency

```
1:  Class InMapperLetterFrequency
2:    Class LetterFrequencyMapper extends Mapper
3:      Variables:
4:        one ← 1
5:        lettersCounter ← empty map
6:
7:      Method setup(context):
8:        Initialize lettersCounter as an empty map
9:
10:     Method map(key, value, context):
11:       data ← convert value to lowercase string
12:       for each character c in data:
13:         if c is a letter:
14:           letter ← strip accents from c
15:           if lettersCounter contains letter:
16:             Increment count of letter in lettersCounter by 1
17:           else:
18:             Add letter to lettersCounter with count 1
19:
20:     Method cleanup(context):
21:       for each entry in lettersCounter:
22:         write (entry.key, entry.value) to context
23:
24:   Class LetterFrequencyReducer extends Reducer
25:     Variables:
26:       totalLetters ← 1.0
27:
28:     Method setup(context):
29:       conf ← get configuration from context
30:       totalLetters ← get double value "totalLetters" from conf, default is 1.0
31:
32:     Method reduce(key, values, context):
33:       tot ← 0.0
34:       for each val in values:
35:         tot ← tot + val
36:       result ← tot / totalLetters
37:       write (key, result) to context
```

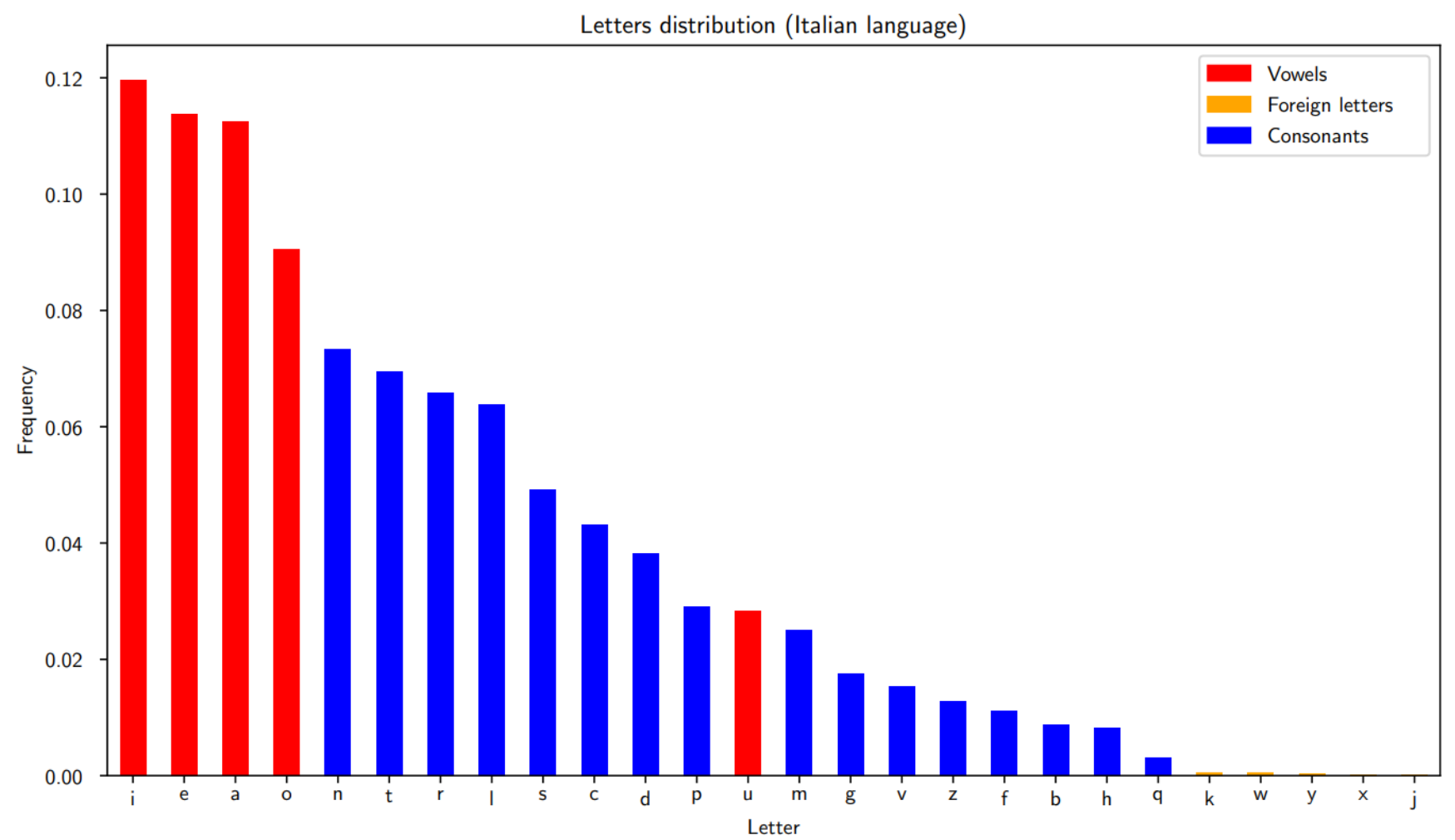
→ *hashMap*

→ *cleanup method*

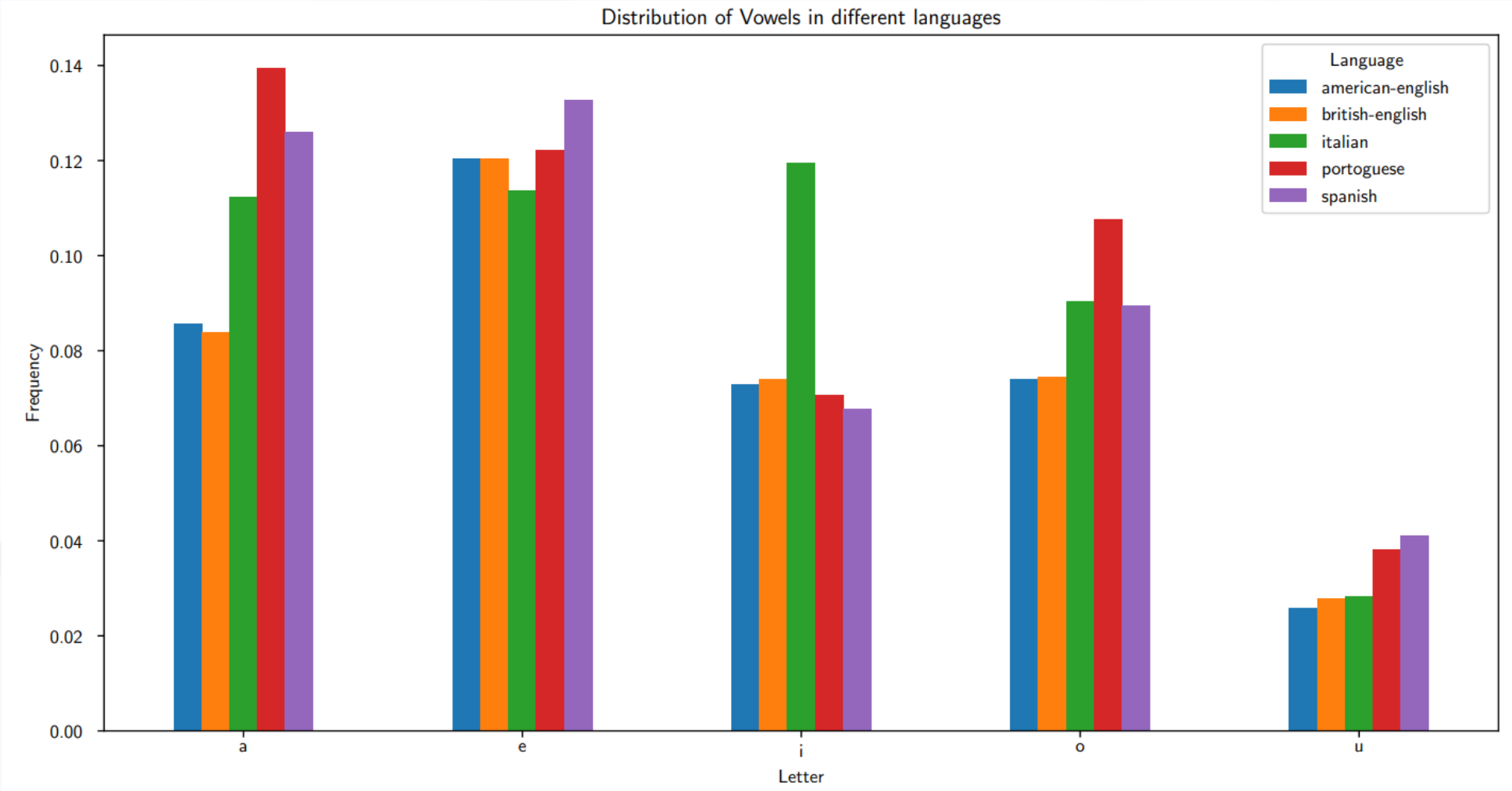
Data analysis
Adopted datasets

- **Articles from newspapers**
- **Languages**
  - English
    - *The New York Times (American english)*
      271KB, 4.9MB, 147MB, 245MB, 489MB, 1.19GB, 2.37GB ← *Performance analysis*
    - *The Guardian (British english)*
      263MB
  - Italian
    - Gonews, 255MB
  - Spanish, 350MB
  - Portoguese, 338MB

Data analysis
Letter distribution (Italian language)

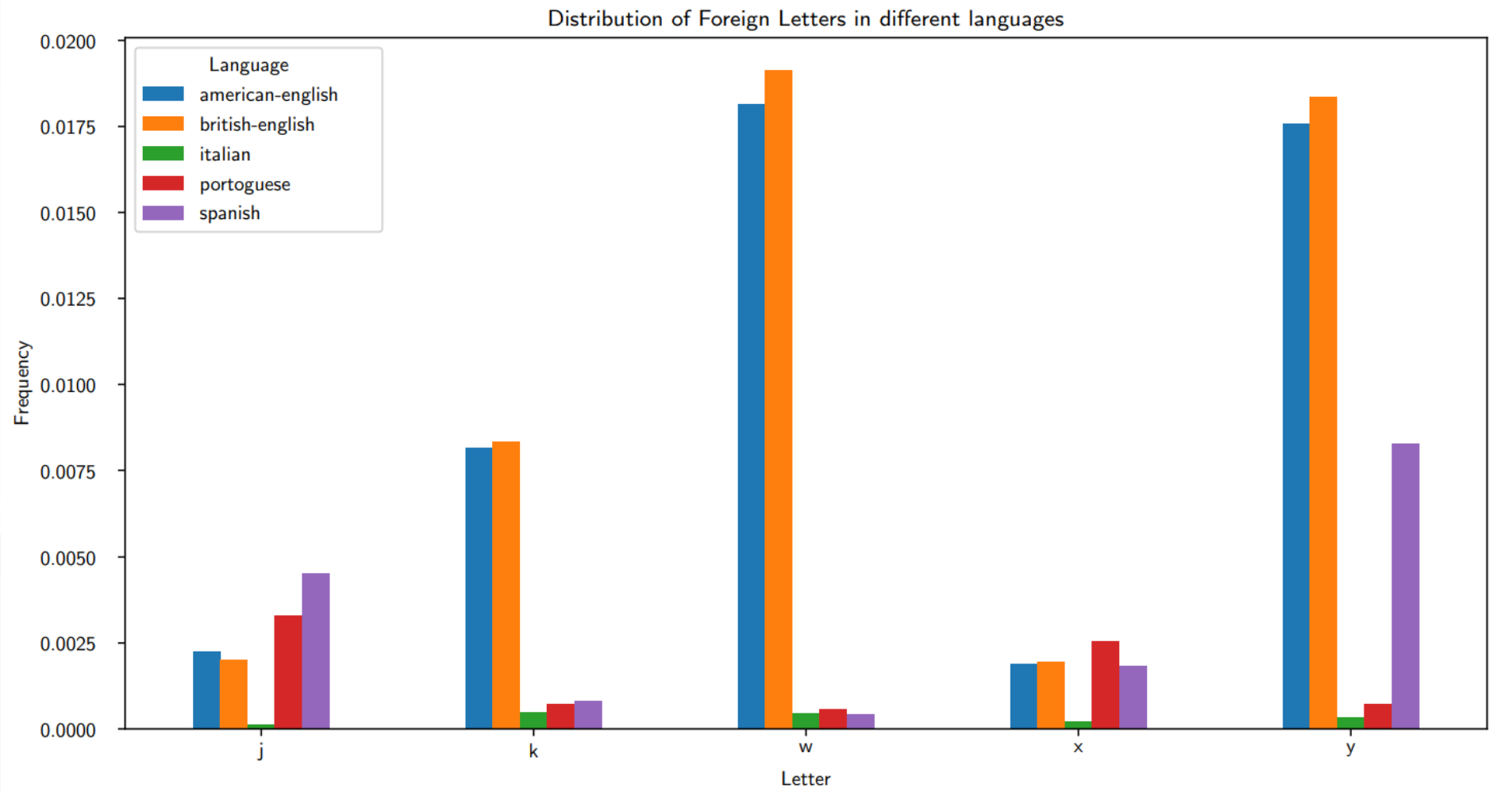


Data analysis
Distribution of Vowels in different languages



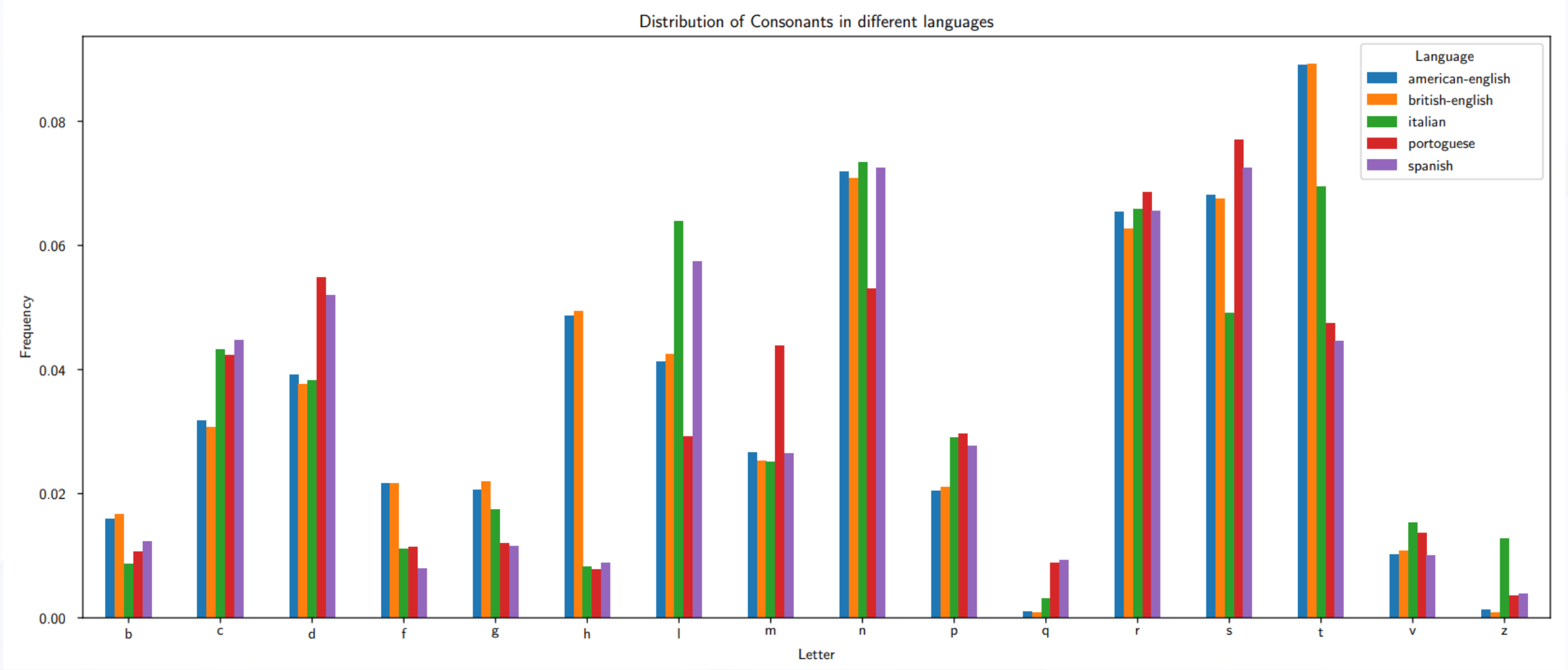
Analyze letter frequency through Apache Hadoop

Data analysis
Distribution of foreign letters in different languages



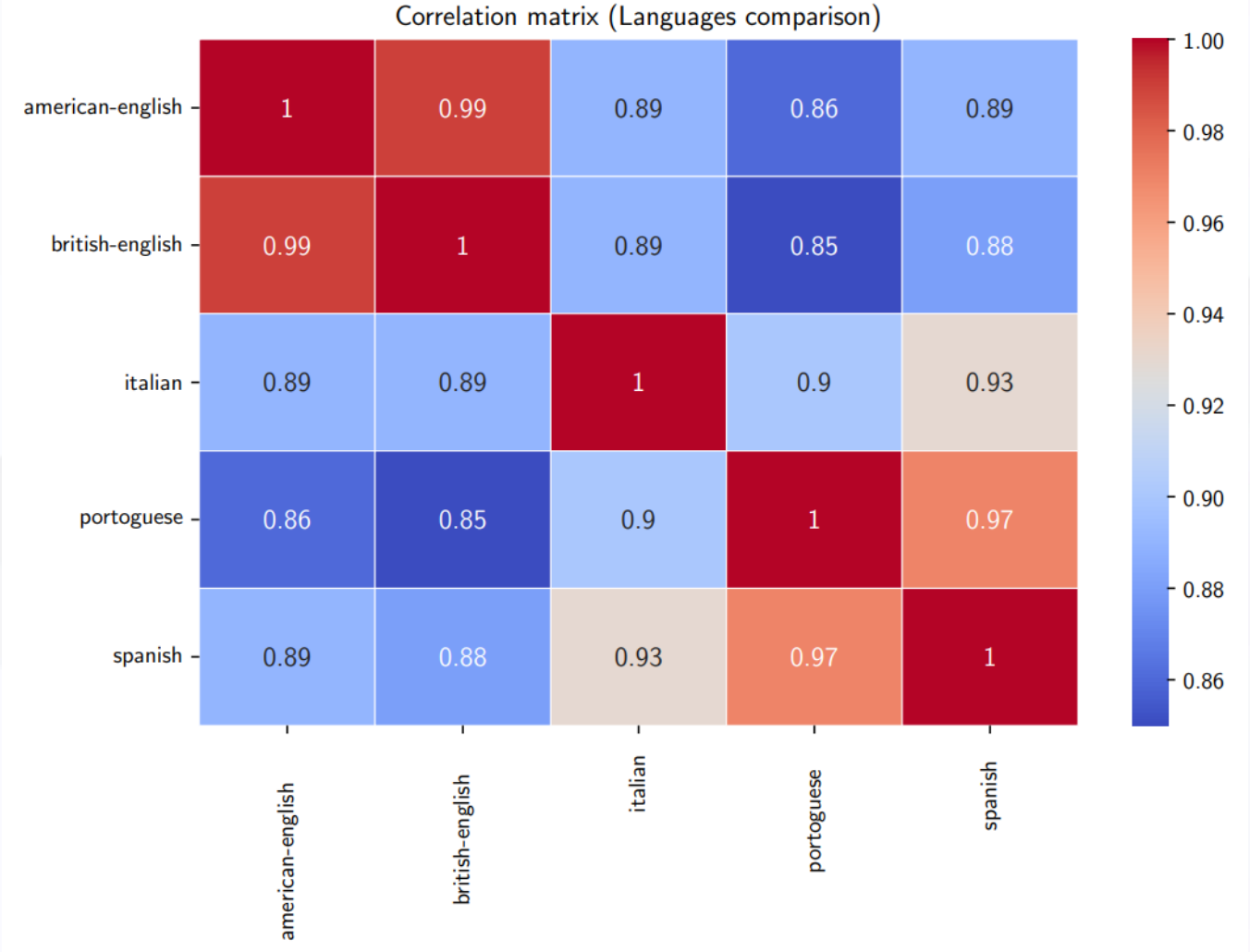
Analyze letter frequency through Apache Hadoop

Data analysis
Distribution of consonants in different languages

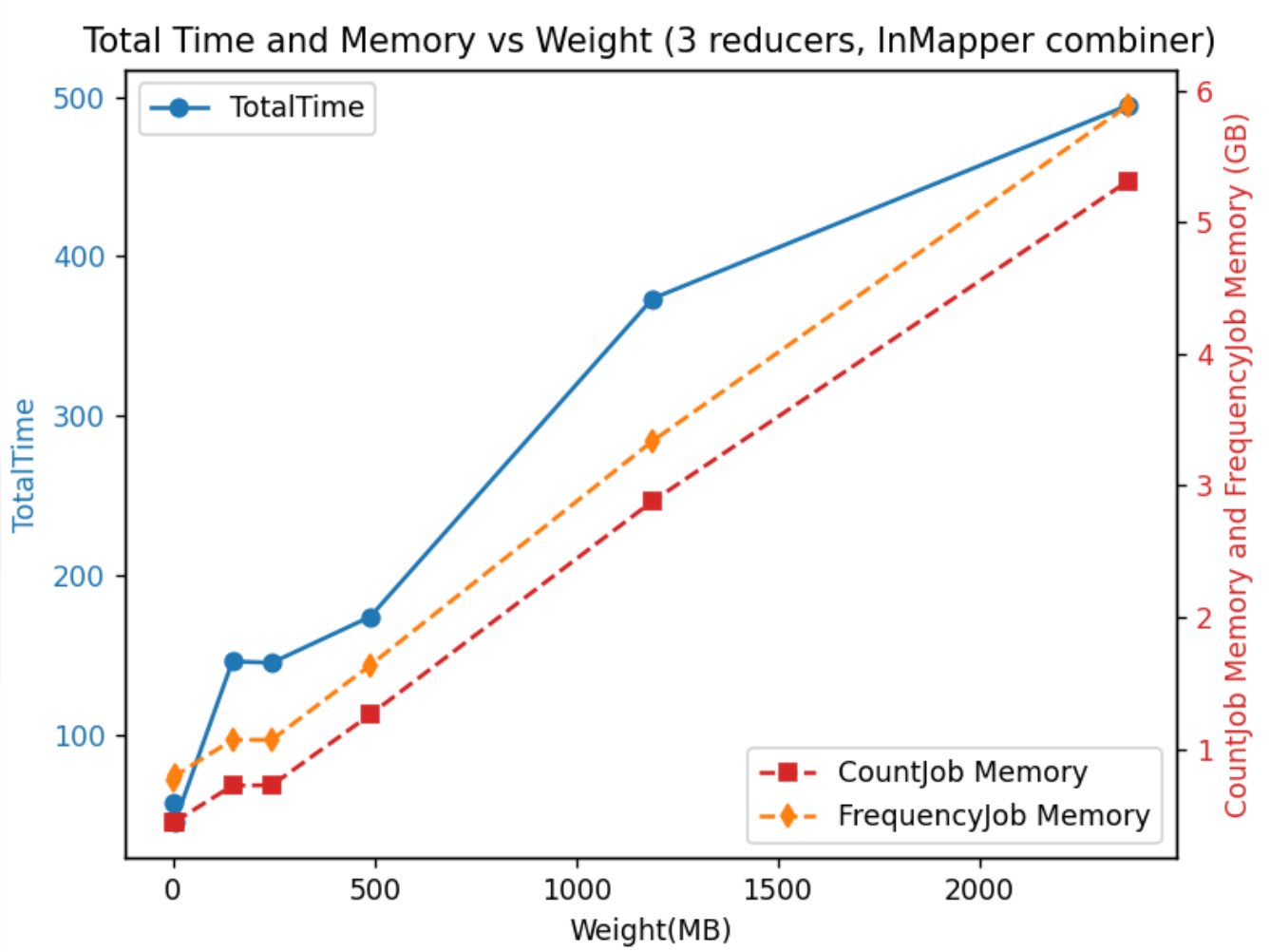


Analyze letter frequency through Apache Hadoop

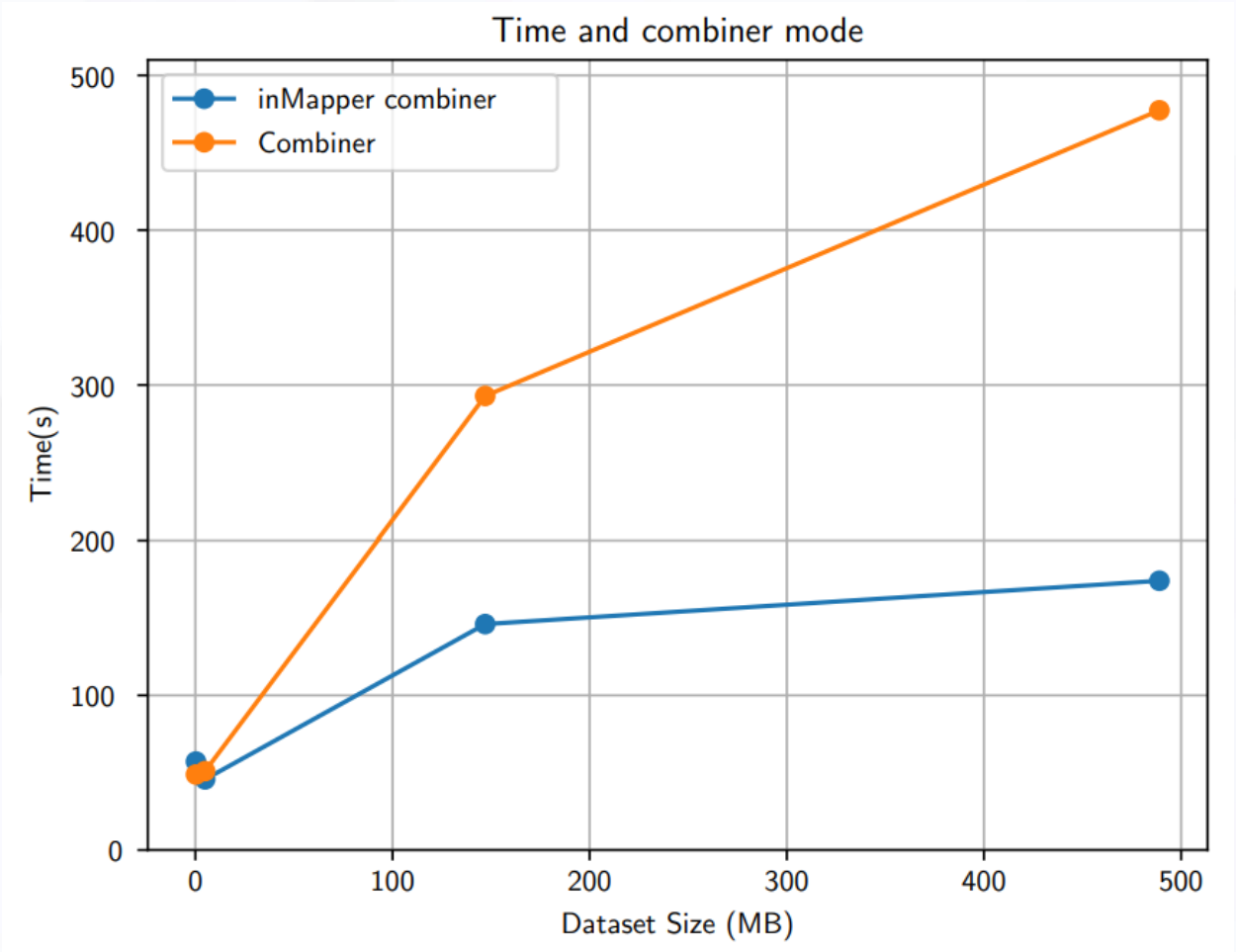
Data analysis
Correlation matrix (Languages comparison)



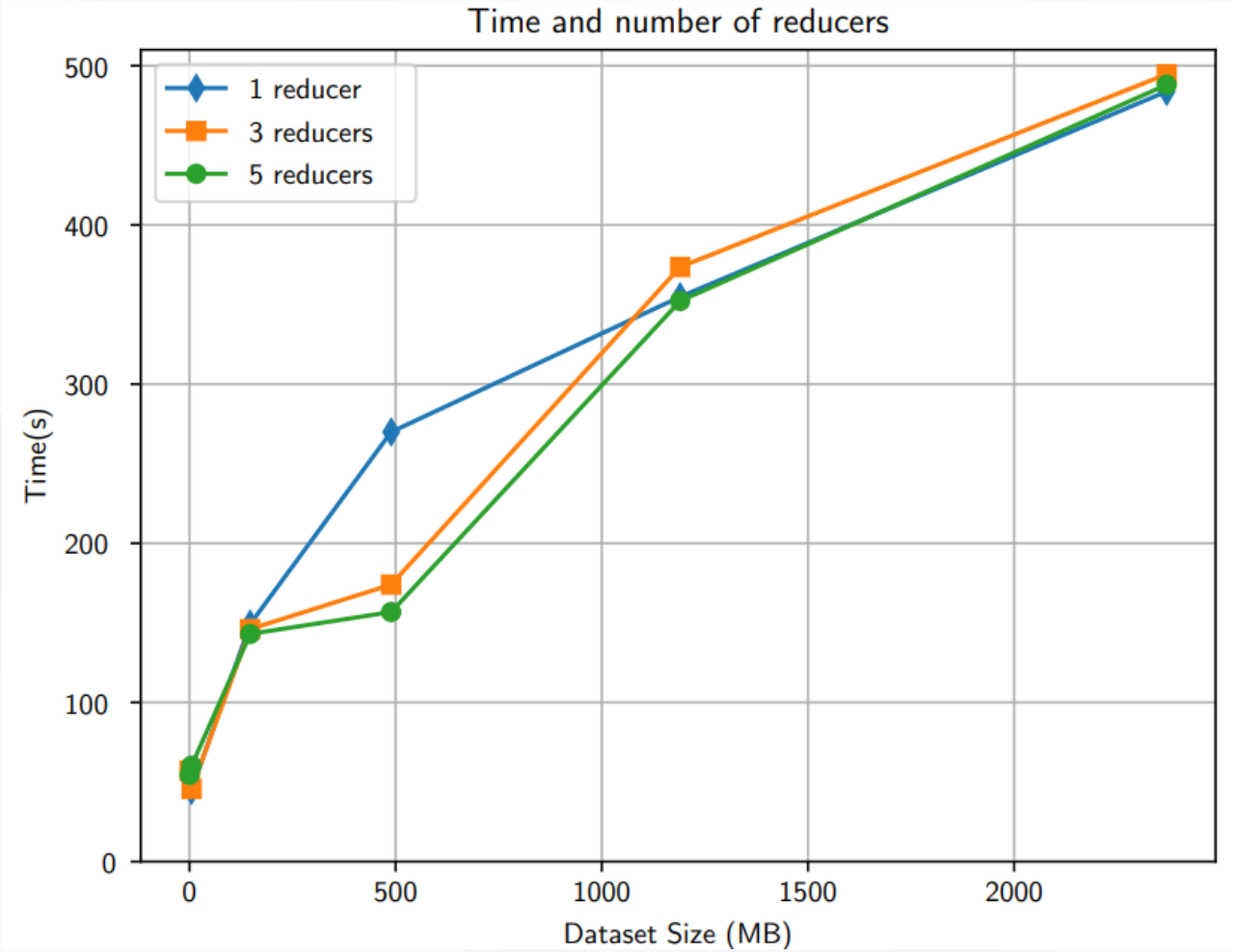
Data analysis
Total time and Memory vs Weight (3 reducers, inMapper combiner)



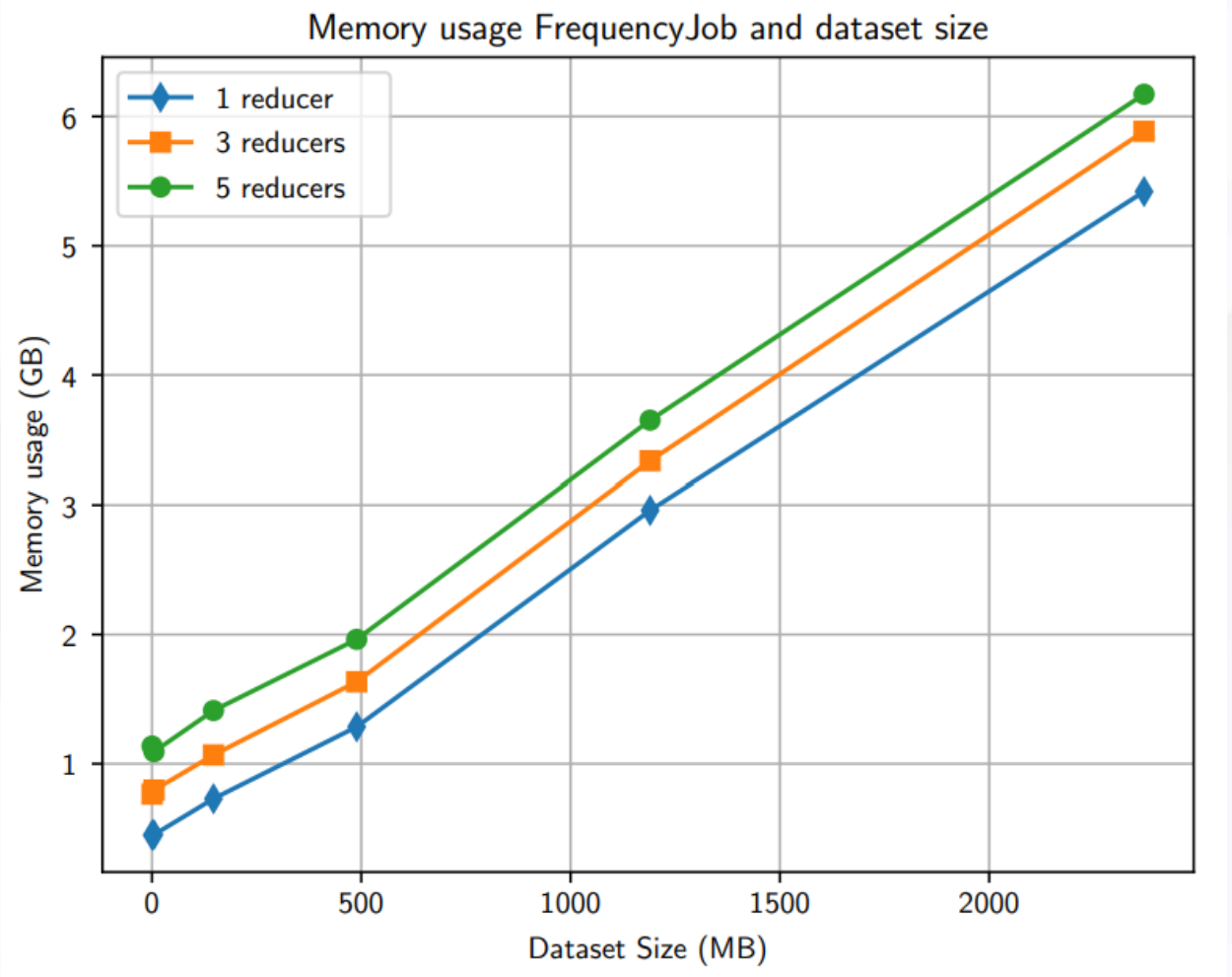
Data analysis
Time and combiner mode



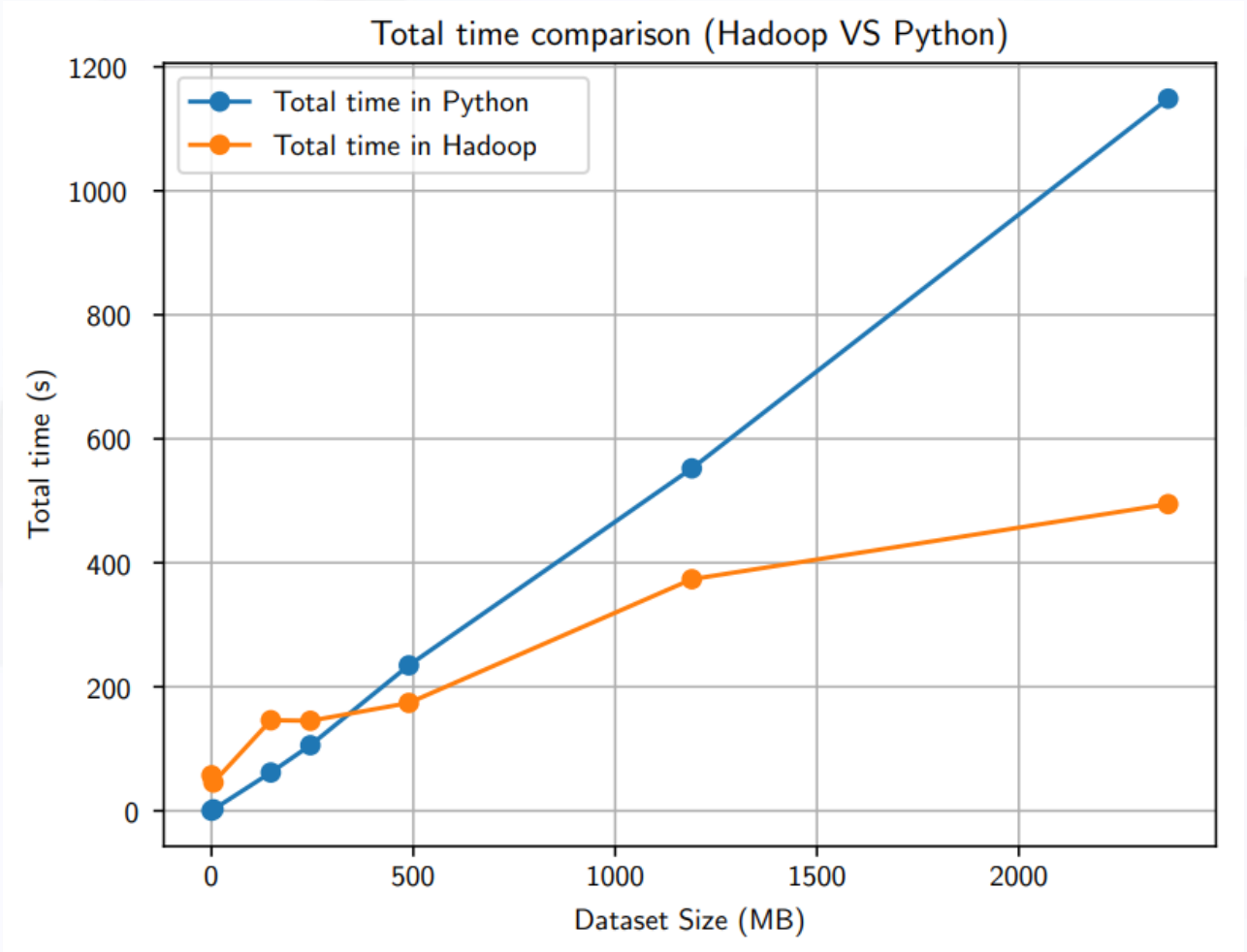
Data analysis
Time and number of reducers



Data analysis
Memory usage FrequencyJob and dataset size



Data analysis
Total time comparison (distributed vs non-distributed approaches)



Thanks!