LETTER FREQUENCY ANALYSIS THROUGH HADOOP

Martina Fabiani
Rossana Antonella Sacco
Tommaso Falaschi

LETTER FREQUENCY ANALYSIS THROUGH HADOOP

LETTER FREQUENCY

Two different MapReduce design patterns:

- Combiner
- InMapping Combining

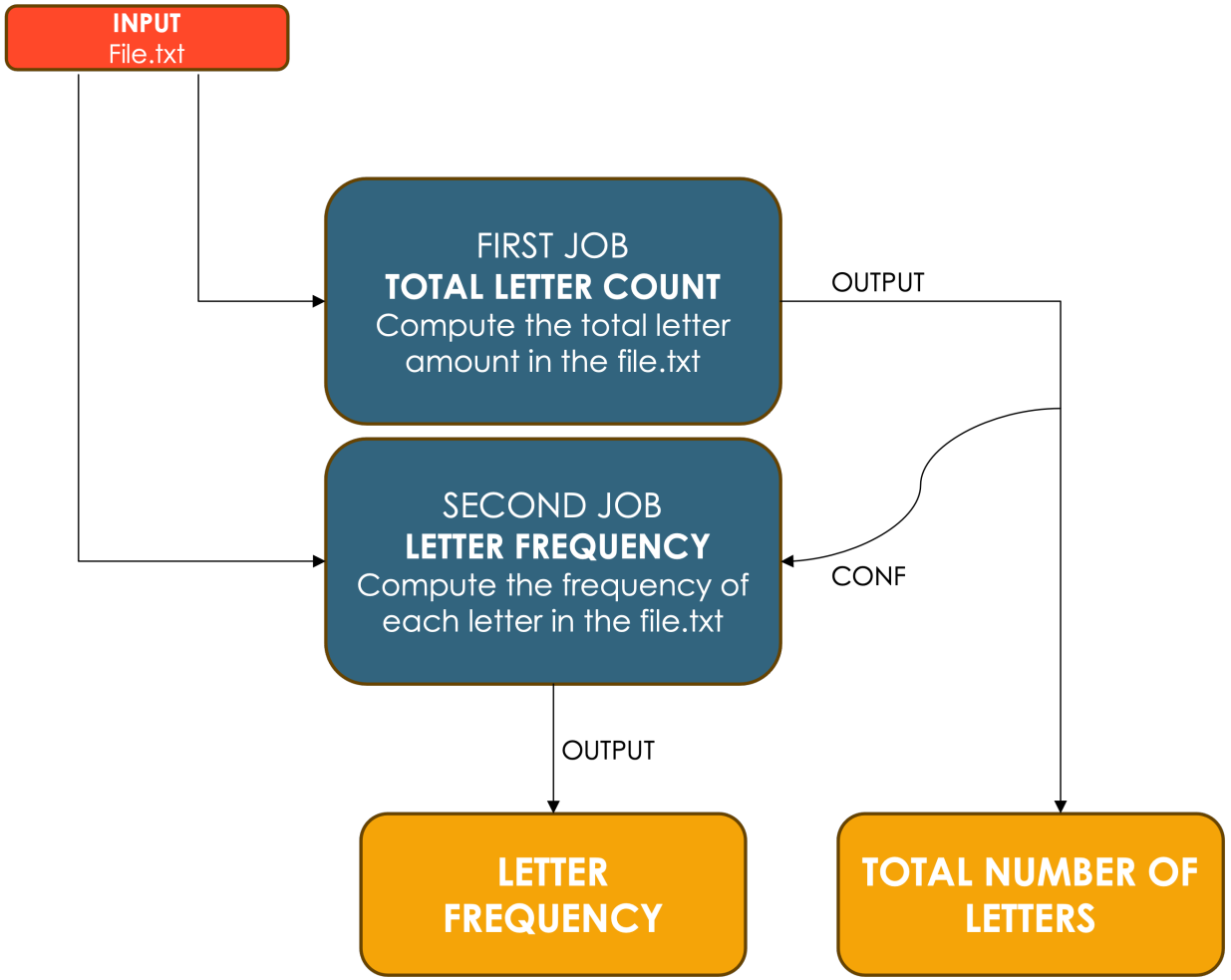
PERFOMANCE ANALYSIS

MapReduce alghoritm execudes with different configuration.

STATISTIC ANALYSIS

Analysis of letter frequency in diffent leanguages, similarites and differences

GENERAL WORKFLOW



COMBINER
TOTAL LETTER COUNT

MAPPER

```
Class MAPPER
1:  method Map(docid a, doc value)
2:      str ← CleanText(value)      ▷ Remove non-letters, accents and set lowercase
3:      for each character c in str do
4:          EMIT(id⊥, count 1)
```

COMBINER
TOTAL LETTER COUNT

REDUCER

```
Class REDUCER
1:  method Reduce(id⊥, counts[n1,n2...])
2:      sum ← 0
3:      for each count n in counts[n1, n2,...] do
4:          sum ← sum + n
5:      EMIT(id⊥, count sum)
```

COMBINER
LETTER FREQUENCY

MAPPER

```
Class MAPPER
1:  method Map(docid a, doc value)
2:      str ← CleanText(value)      ▷ Remove non-letters,accents and set lowercase
3:      for each character c in str do
4:          EMIT(character c, count 1)
```

COMBINER
LETTER FREQUENCY

COMBINER

```
Class COMBINER
1:  method Combine(character c, counts[n1,n2...])
2:      sum ← 0
3:      for each count n in counts[n1, n2,...] do
4:          sum ← sum + n
5:      EMIT(character c, count sum)
```

COMBINER
LETTER FREQUENCY

REDUCER

```
Class REDUCER
1:  method Initialize(Context context)
2:       TOTAL_LETTERS ← GetTotalLetters(context) ▷ Retrieve configuration
3:  method Reduce(character c, counts [n1, n2,...])
4:       sum ← 0
5:       for each count n in counts [n1, n2,...] do
6:            sum ← sum + n
7:       freq ← sum / TOTAL_LETTERS
8:       EMIT(character c, frequency freq)
```

INMAPPING COMBINING
TOTAL LETTER COUNT

MAPPER

```
Class MAPPER
1:  method Initialize
2:      sum ← 0
3:  method Map(docid a, doc value)
4:      str ← CleanText(value)      ▷ Remove non-letters,accents and set lowercase
5:      for each character c in str do
6:          sum ← sum + 1
7:  method Close
8:      EMIT(id⊥, count sum)
```

INMAPPING COMBINING
TOTAL LETTER COUNT

REDUCER

```
Class REDUCER
1:  method Reduce(id⊥, counts[c1,c2...])
2:      sum ← 0
3:      for each count c in counts[c1, c2,...] do
4:          sum ← sum + c
5:      EMIT(id⊥, count sum)
```

INMAPPING COMBINING
LETTER FREQUENCY

MAPPER

```
Class MAPPER
1:  method Initialize
2:      map ← new ASSOCIATIVEARRAY
3:  method Map(docid a, doc value)
4:      str ← CleanText(value)      ▷ Remove non-letters,accents and set lowercase
5:      for each character c in str do
6:          map{c} ← map{c} + 1
7:  method Close
8:      for each character c ∈ map do
9:          EMIT(c, count map{c})
```

INMAPPING COMBINING
TOTAL LETTER COUNT

REDUCER

```
Class REDUCER
1:  method Initialize(Context context)
2:       TOTAL_LETTERS ← GetTotalLetters(context) ▷ Retrieve configuration
3:  method Reduce(character c, counts [n1, n2,...])
4:       sum ← 0
5:       for each count n in counts [n1, n2,...] do
6:            sum ← sum + n
7:       freq ← sum / TOTAL_LETTERS
8:       EMIT(character c, frequency freq)
```

PERFORMANCE ANALYSIS

DIFFERENT DESIGNS

- COMBINER
- INMAPPING COMBINING

DIFFERENT NUMBER OF REDUCERS

FROM **1** TO **3** REDUCERS

DIFFERENT DATA SIZES

- 2.09 **KB**
- 2.14 **MB**
- 2.15 **GB**

PERFORMANCE ANALYSIS

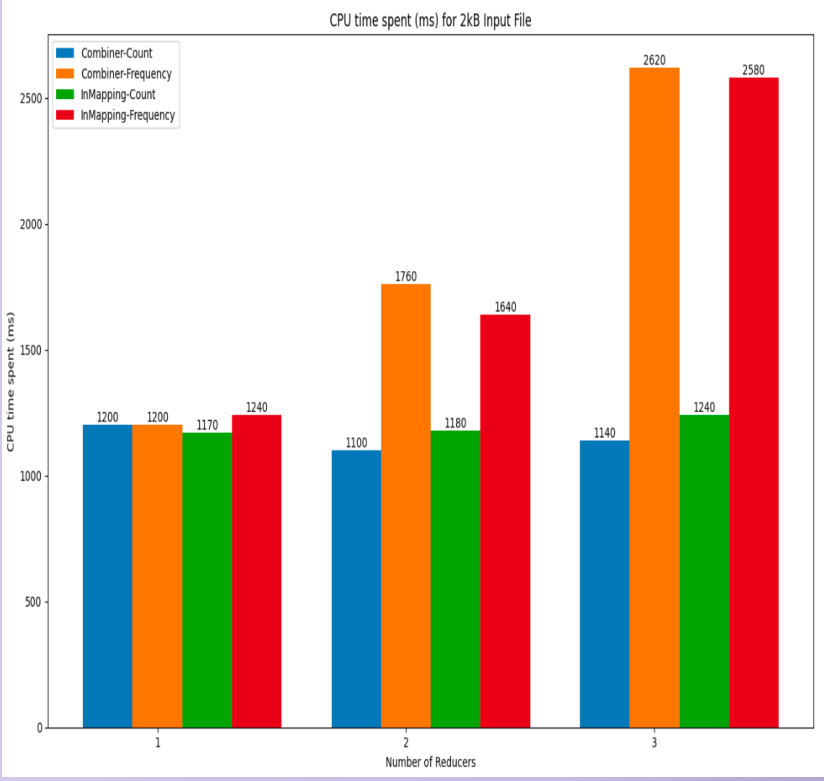
PARAMETER CONSIDERED

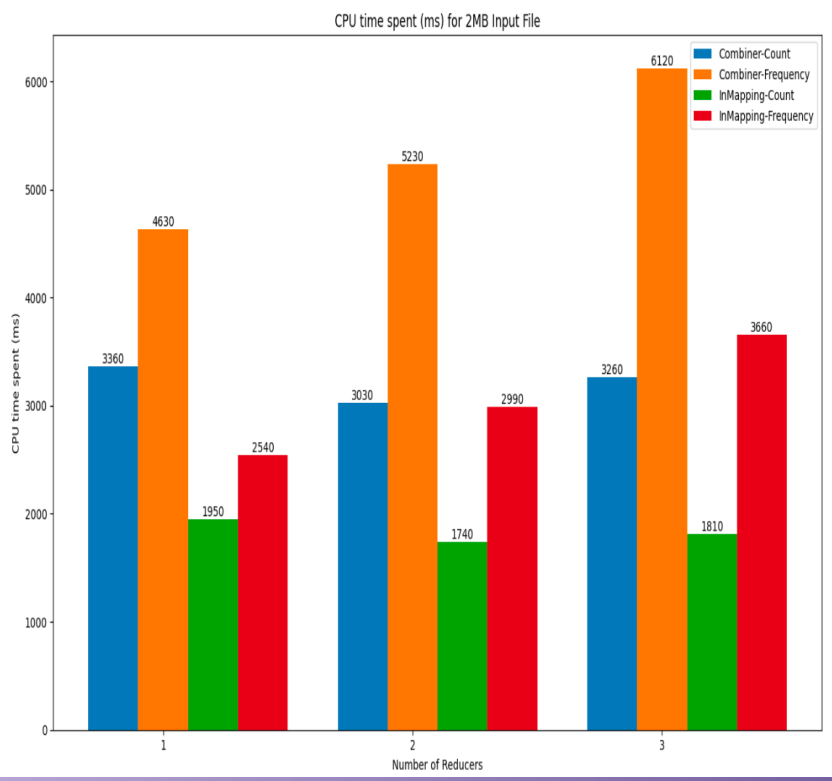
Total time spent by all MAP tasks	Total time spent by all REDUCE tasks	CPU time spent	Garbage Collection (GC) time elapsed	Reduce shuffle bytes	Physical memory (bytes) snapshot

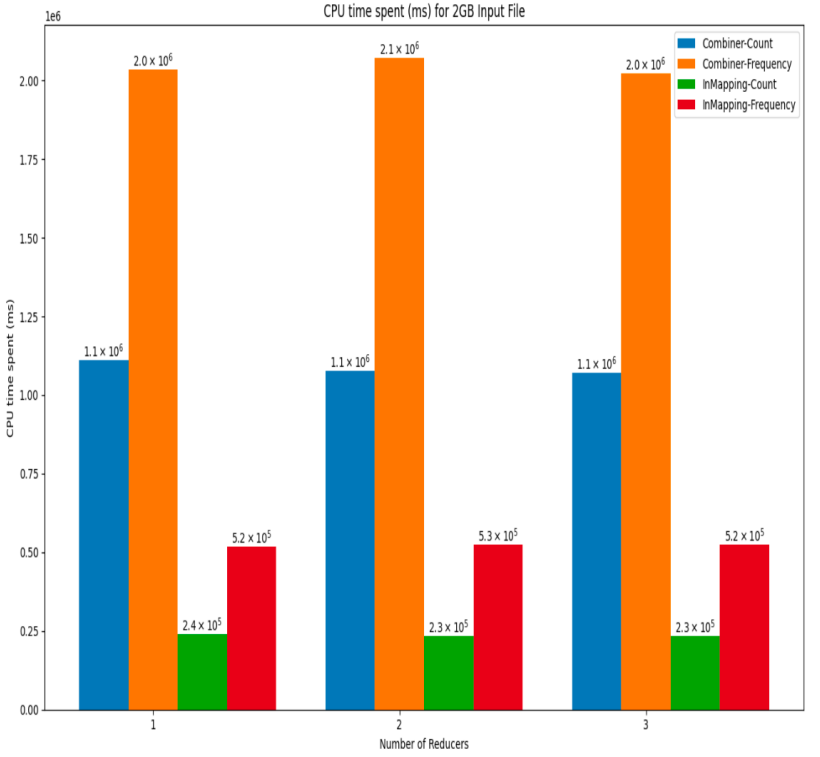
(ms)

PERFORMANCE ANALYSIS

MAIN RESULTS

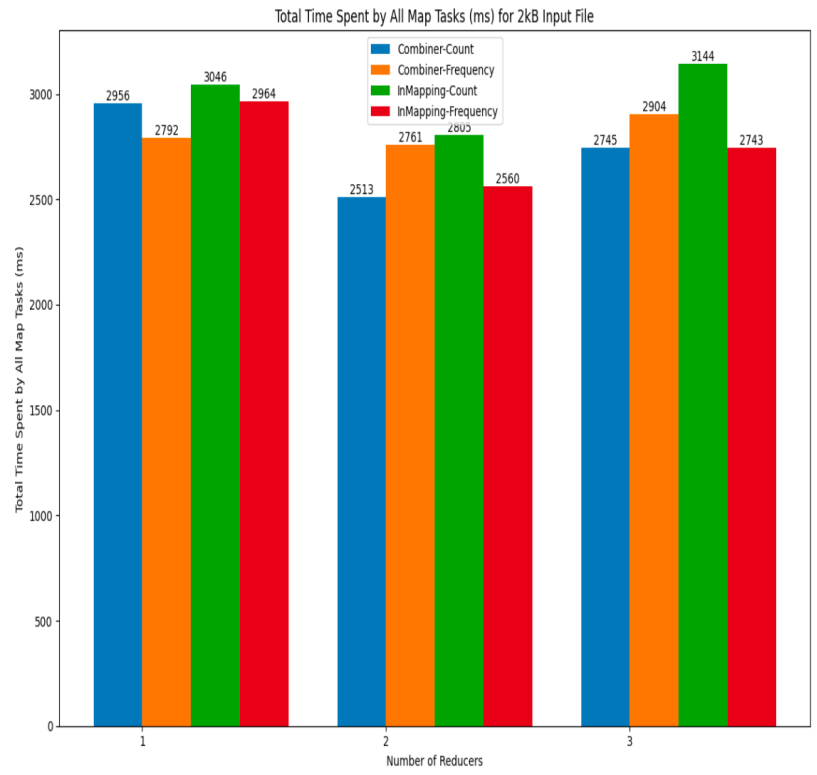


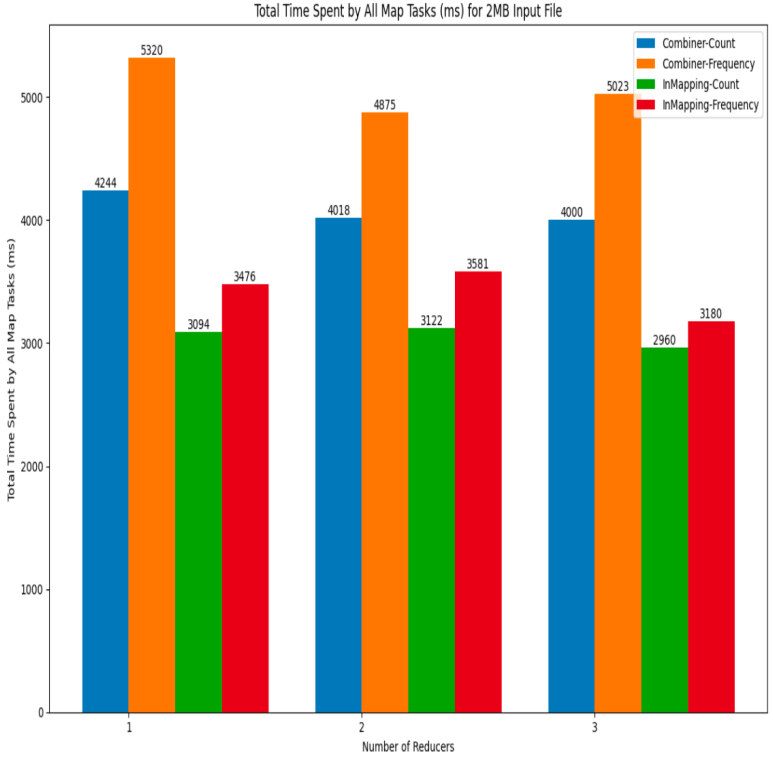


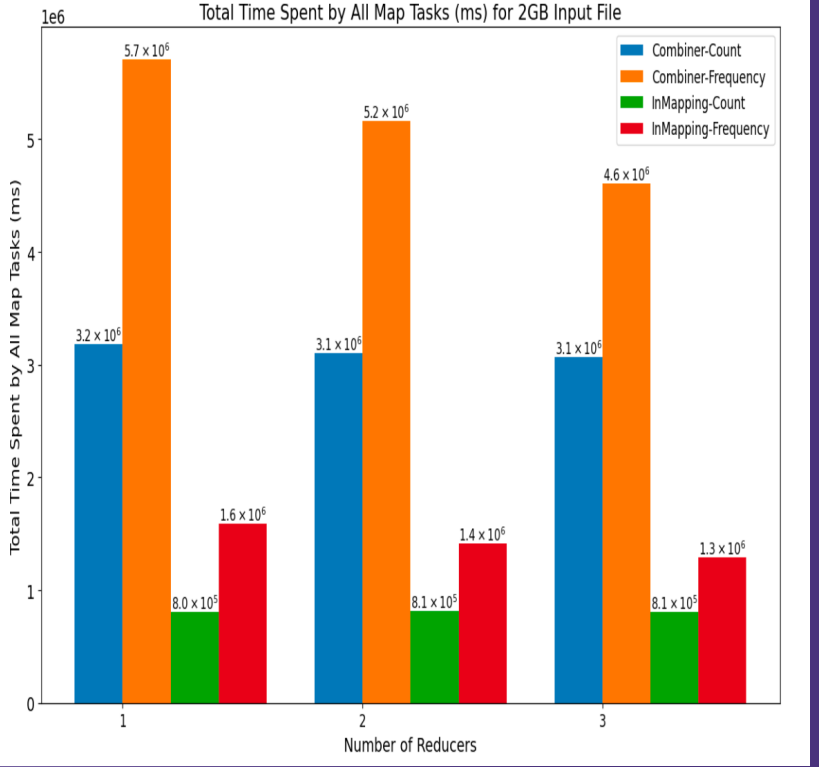


PERFORMANCE ANALYSIS

MAIN RESULTS

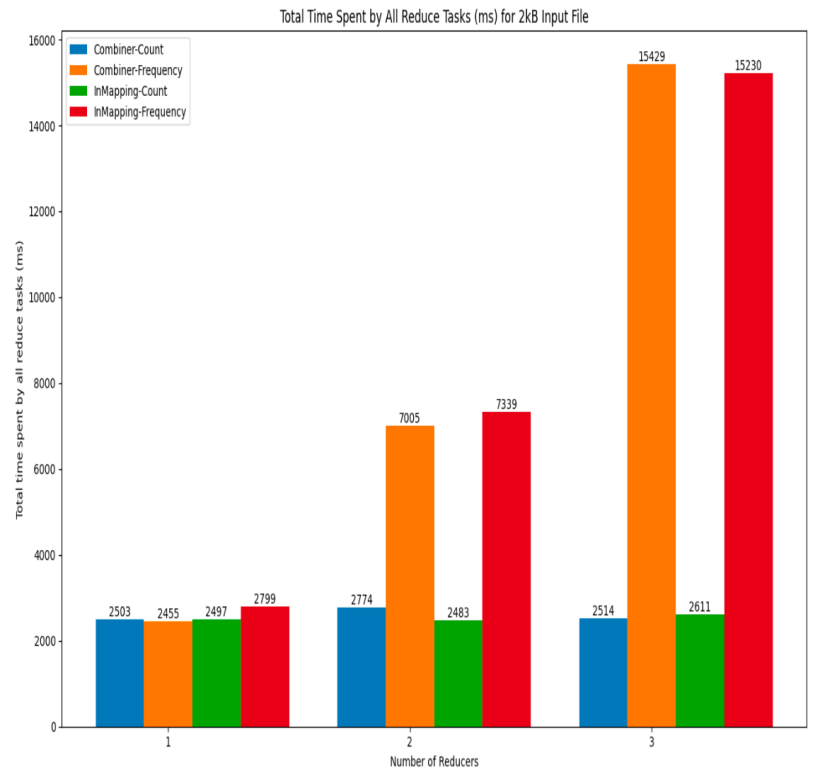


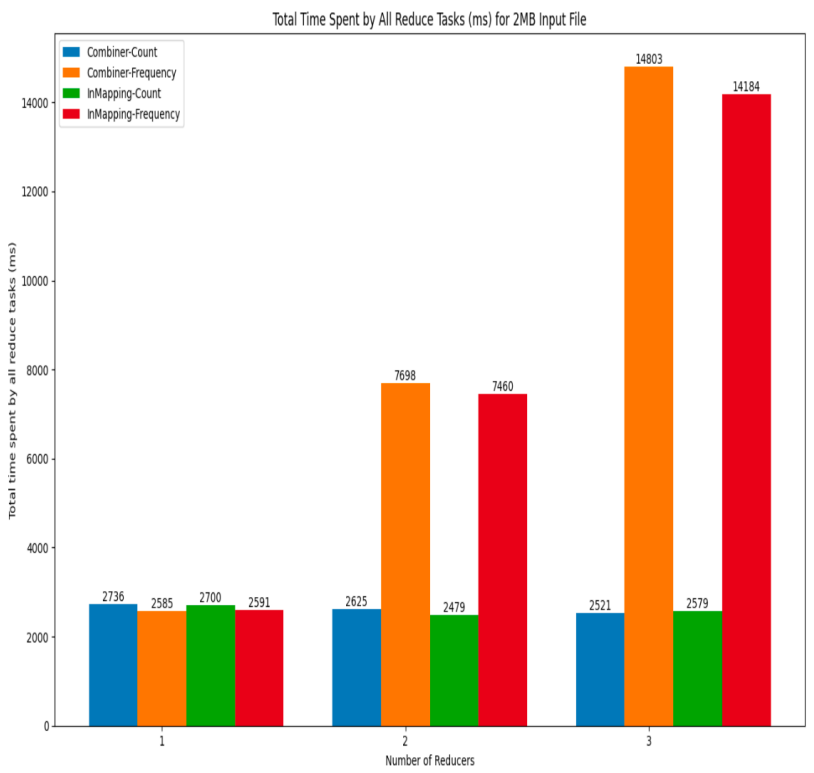


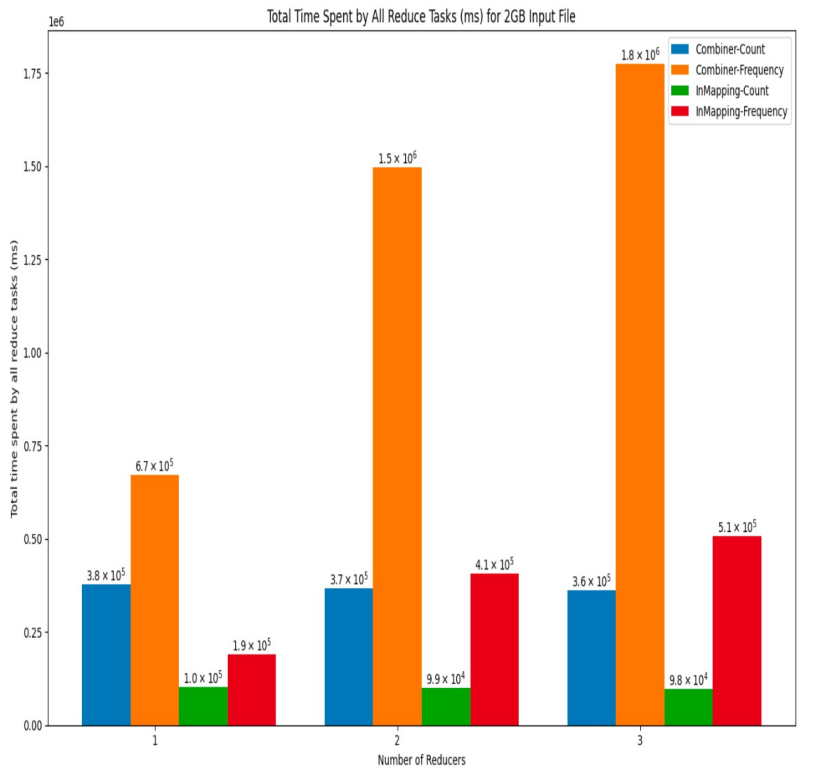


PERFORMANCE ANALYSIS

MAIN RESULTS







PERFORMANCE ANALYSIS

CONSIDERATIONS

BEST DESIGNS

- **COMBINER** is better with little size input
- **INMAPPING COMBINING** is better with large size input

NUMBER OF REDUCERS

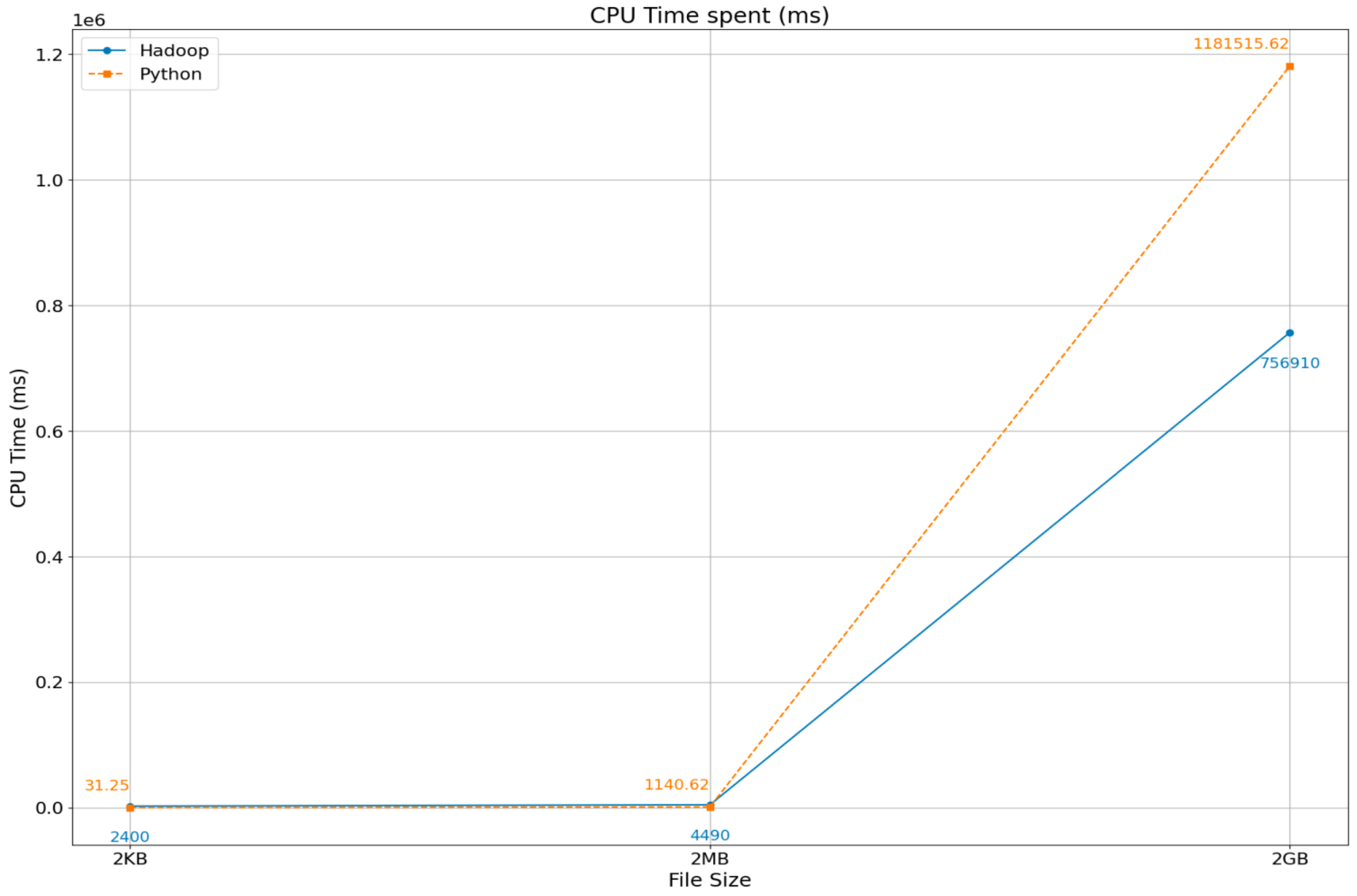
1

PERFORMANCE ANALYSIS

HADOOP VS PYTHON

PYTHON LOCAL EXECUTION

To do this comparison, we've used the same input files as before, and used **CPU time spent** as the benchmark.



STATISTIC ANALYSIS

INPUT FILE.TXT	ALGHORITM	NUMBER OF REDUCER
PINOCCHIO (250 kb)	COMBINER	1

STATISTIC ANALYSIS

DIFFERENT LENGUAGES

FINNO-UGRIC

- FINNISH

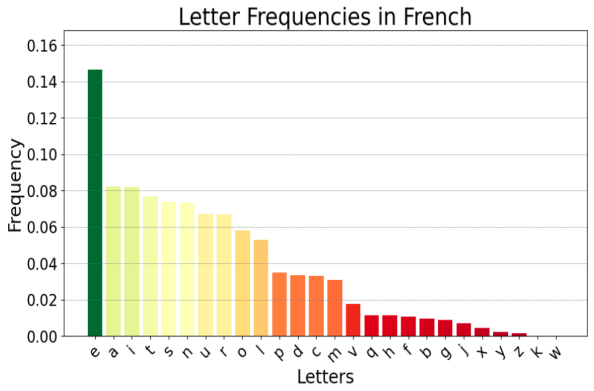
GERMANIC

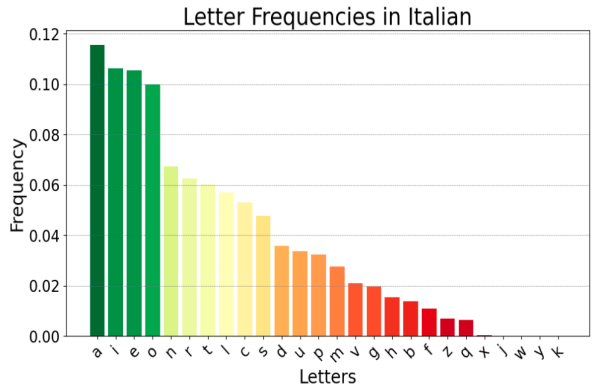
- DUTCH
- ENGLISH
- GERMAN

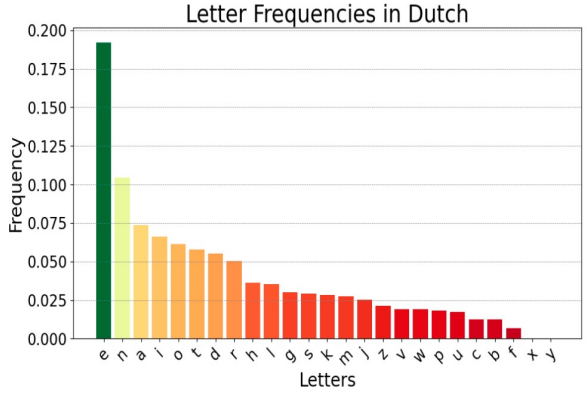
ROMANCE

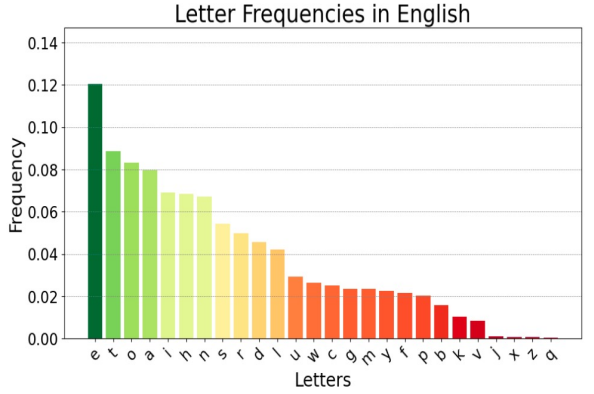
- ITALIAN
- FRENCH
- PORTUGUESE
- SPANISH

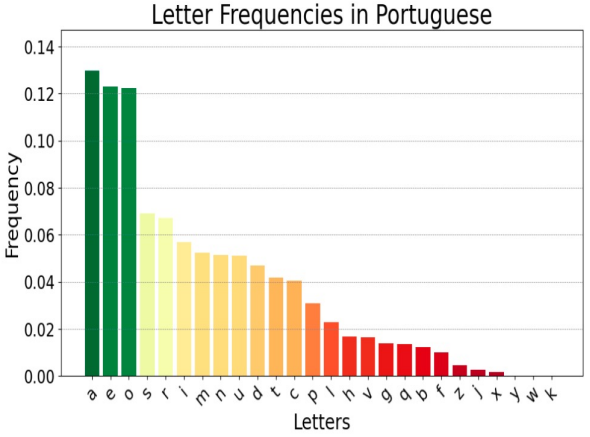
STATISTIC ANALYSIS
MAIN RESULTS

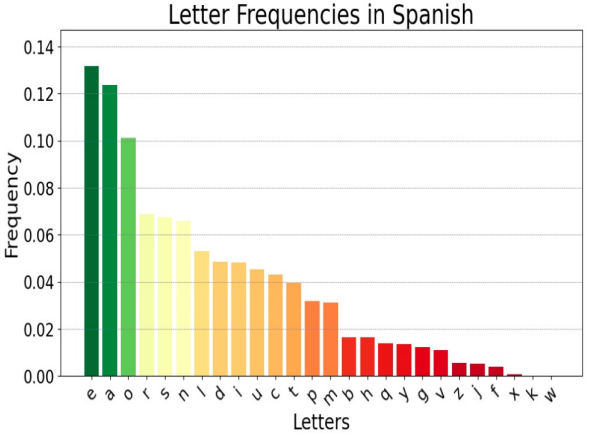


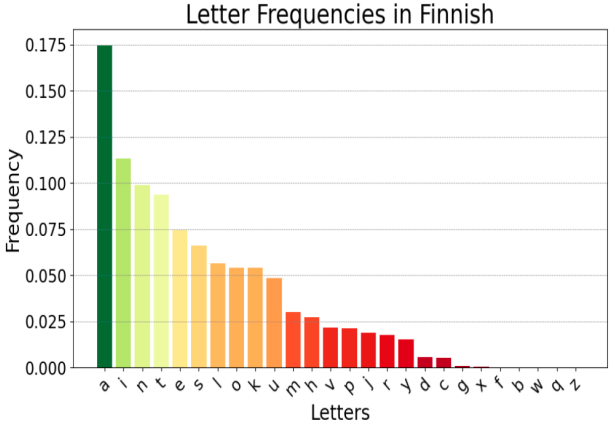


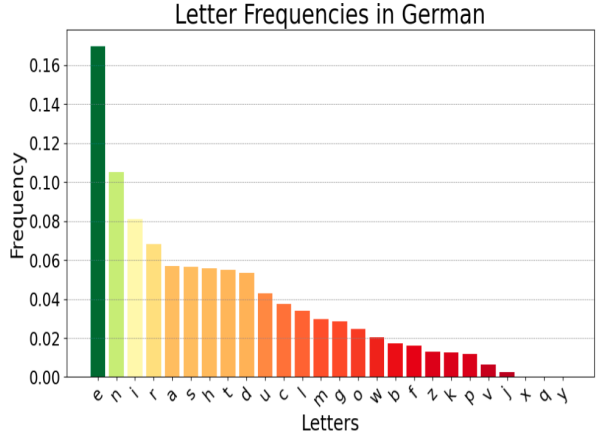




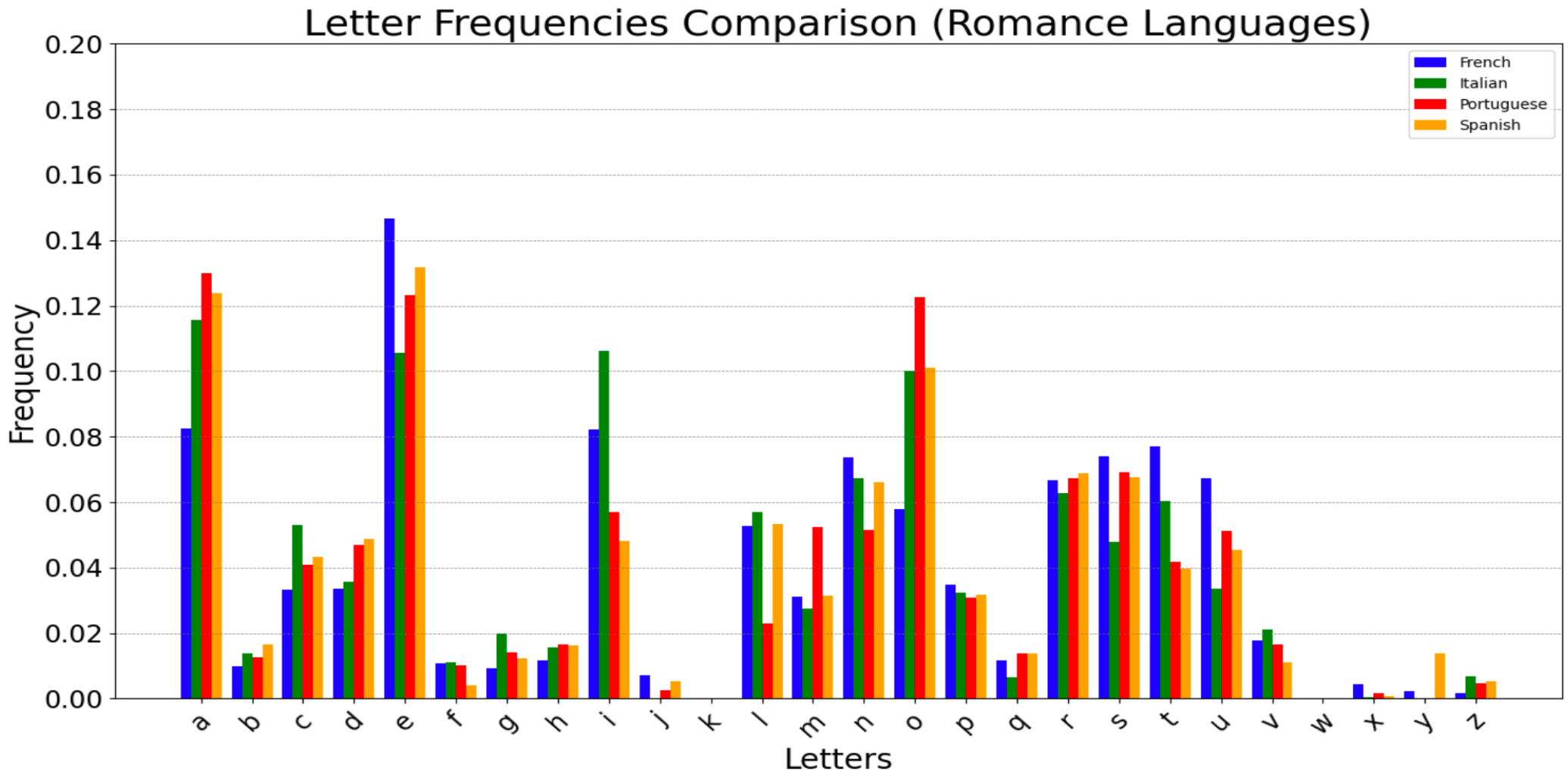




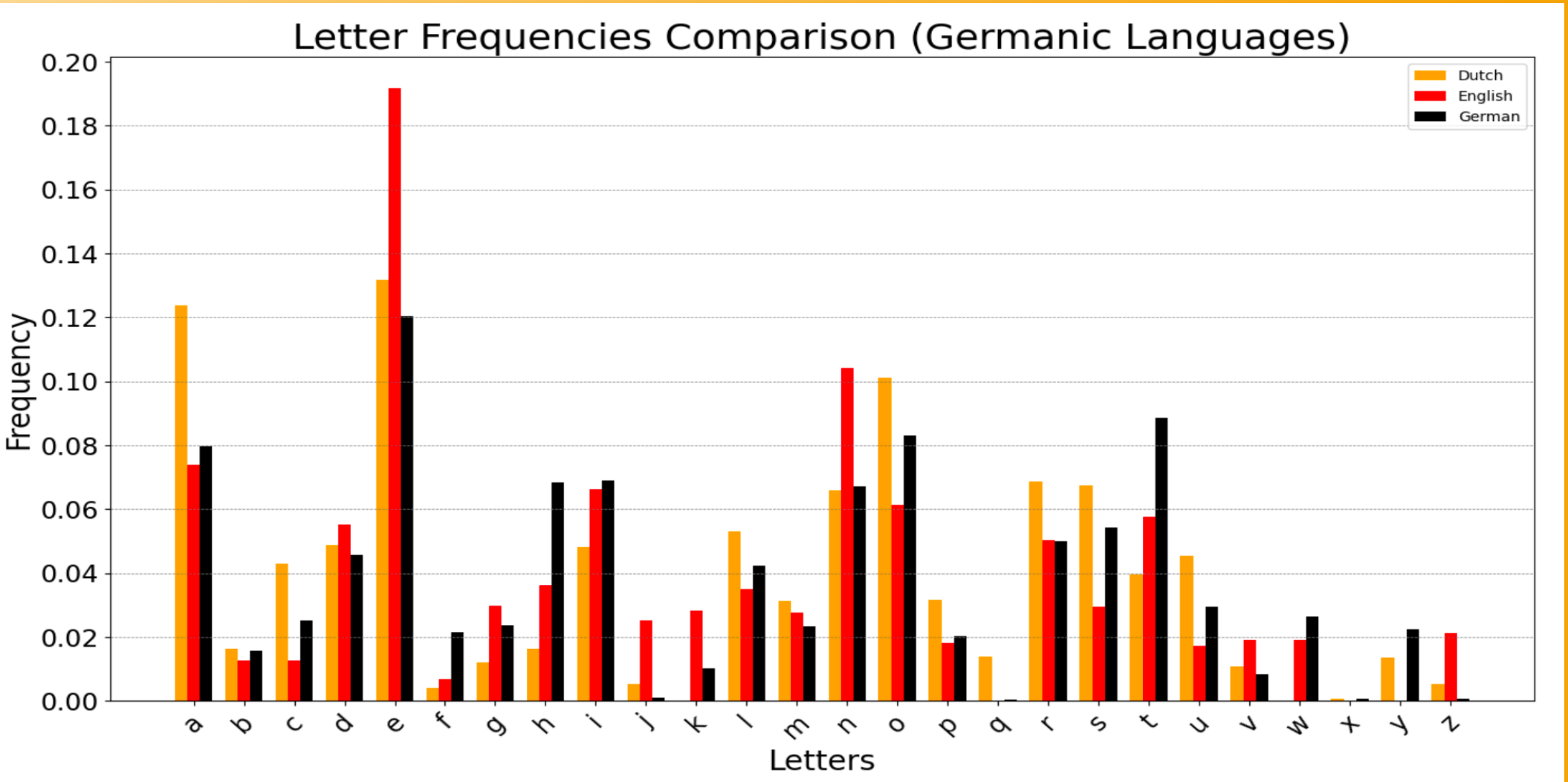




STATISTIC ANALYSIS
MAIN RESULTS



STATISTIC ANALYSIS
MAIN RESULTS



THANK YOU!

- MARTINA FABIANI
- ROSSANA ANTONELLA SACCO
- TOMMASO FALASCHI