Project 3: Frequent Items Counting
A Study on Memory-Efficient Algorithms

Filipe Pires [85122] & Joo Alegria [85048]

Advanced Algorithms
Department of Electronics, Telecommunications and Informatics
University of Aveiro

Abstract **– Lorem ipsum ...**

Keywords **– Probabilistic Counter, Count-Min Sketch, Memory-Efficient Algorithms**

I. Problem Contextualization

This report was written for the course of 'Advanced Algorithms', taught by professor Joaquim Madeira for the master's in Informatics Engineering at DETI UA. It describes the work done for the third assignment of the course [*1*]. The chosen hypothesis was "Hiptese A-2 Contagem dos Itens Mais Frequentes".

One of the subjects studied in this course was the counting problems and the obstacles that appear when dealing with this type of problems in very large contexts. The difficulties appear even more when the contexts morph into the newest, fastest and the form that delivers the most amount of information, the data streams. Being a continuous source of real time events, processing the events that happen in a precise and reliable way can be more ofter than not very challenging and sometimes impossible. Many solutions appear in hopes of creating reliable approximations that without using the as many resources still permit inferring the same conclusions. This solutions use many times probabilistic approaches or some kind of mapping that allows some form of overlapping of the data without compromising the results, such as hashing.

This theme was already a subject for previous works, but since it is an area so vast and with so many different approaches, each one with its advantages and disadvantages, the professor decided to list it again for the possible report themes in this last project, but this time with more complex and refined algorithms that use memory and resources in a more intelligent way. We ended up choosing this theme since we are interested in knowing how lower can the resource usage go without compromising the outcome of the model, meaning that the conclusions taken from the approximations made by the model are not too distant from the ones taken from the exact counters.

In the context of literary works as source of text, we create data streams to simulate a real world scenario and test the effectiveness and performance of one of the proposed algorithms. Some of those proposed algorithms were the one from Misra & Gries(Frequent-Count), the one from Manku & Motwani(SLossy-Count), the one from Metwally(Space-Saving-Count) and lastly the Count-Min Sketch. We choose to analyze the last one, algorithm that will be further described in Section III. As a extra task, because the dataset are books, if using their translations, we can also discuss if the most common words in one language are the as the ones found in another, which we can expect to be similar since the story is the same.

II. Dataset

The dataset used, as already mentioned in Section I, is based on literary works and in some of their translations. All of the books were obtained from *Project Gutenberg* [*2*]. The chosen books and respective idioms are presented below:

- *Alice in the Wonderland*, by Lewis Carol - written in English, German, French and Italian
- *A Christmas Carol*, by Charles Dickens - written in English, Finnish, German, Dutch and French
- *King Solomon's Mines*, by H. Rider Haggard - written in English, Finnish and Portuguese
- *Oliver Twist*, by Charles Dickens - written in English, French and German
- *The Adventures of Tom Sawyer*, by Mark Twain - written in English, Finnish, German and Catalan

Since provident from the *Project Gutenberg*, the books are provided with some headers and annotations that identify the author, the source and some additional metadata that was not relevant for our study; for that reason we proceeded to execute a quick preprocessing over the various book files to remove that metadata that could cause unwanted variation of results. This process was quite simple and was only necessary the usage of a text editor and the manipulation of some regular expressions.

III. Frequent Word Identification

Lorem ipsum ...

A. Exact Counter

Lorem ipsum ...

B. *Count-Min Sketch*

Lorem ipsum ...

IV. Literary Study

As a form of automation and to enable us to quickly study the behavior of the chosen algorithm when changing some of the necessary parameters for the correct functioning of the program we decided to create a script in Python3 that resolved all our problems. The signature of that script is as follows:

```
python3 frequentWordFinder.py [-h]
[-o outputFolder] [-d numHash]
[-m numColumns] <inputFolder>
```

Where the *outputFolder* is a optional parameter that indicates the folder where the resulting files should be stored, the *numHash* is also an optional parameter and is the number of hashes the user wants for the script to use when creating the `CountMinSketch` object; *numColumns* is also optional and is the number of columns used also when creating the `CountMinSketch` object, which as already explained in III, is the space allocated to when storing the hashing results, and the bigger the better since that will cause less conflicts. Lastly the input folder is a mandatory parameters since it's the folder where the source data should be present; this data should be already preprocessed, since that will generate the best results, without any noise form the headers and annotations mentioned in II.

V. Results & Discussion

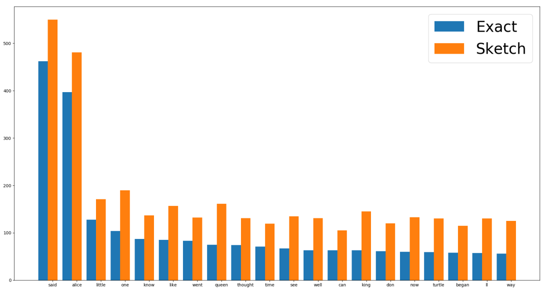
Lorem ipsum ...

VI. Conclusion

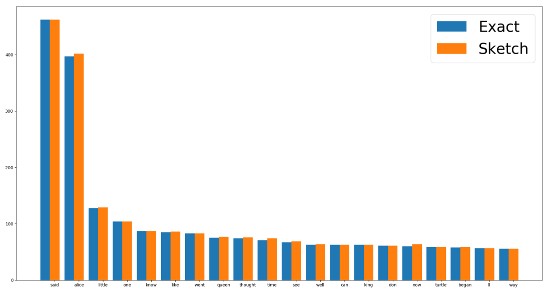
Lorem ipsum ...

References

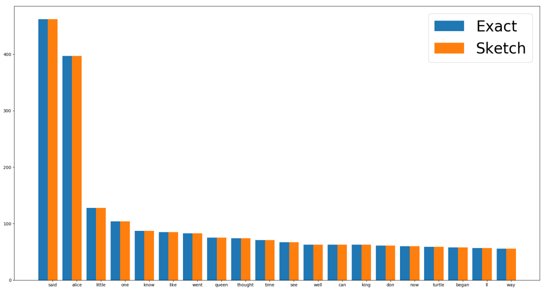
1. Joaquim Madeira, "3o trabalho - algoritmos probabilsticos", `https://elearning.ua.pt/pluginfile.php/1514391/mod_resource/content/0/AA_1920_Trab_2.pdf`.
2. Michael Hart, "Free ebooks - project gutenberg", `https://www.gutenberg.org/`.



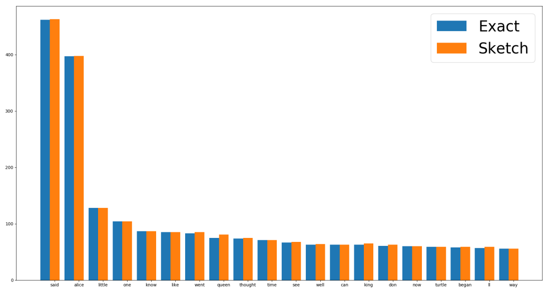
(a) Lorem ipsum



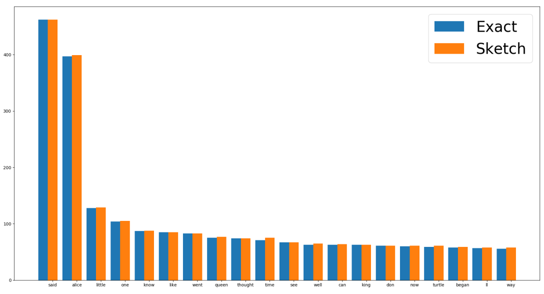
(b) Lorem ipsum



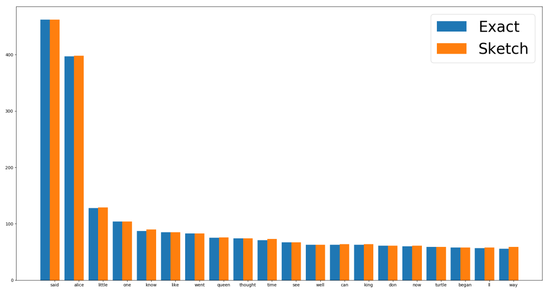
(c) Lorem ipsum



(d) Lorem ipsum



(e) Lorem ipsum



(f) Lorem ipsum

Fig. 1: Caption place holder