Project 3: Using Count-Min Sketch to Find Heavy-Hitter Words

Tiago Fernandes, 88784

***Abstract* – In this project, a probabilistic data structure, the Count-Min Sketch, was used to solve the heavy-hitters problem using strings of text. The quality of this method's outputs was evaluated in relation to the number of false positives and the root mean squared error between the real and estimated counts of the words. It was found that the performance of the algorithm was related to the initialisation parameters, m and d. In fact, the metrics decreased super-exponentially with the increase of each parameter. This meant that very satisfactory results could be achieved using much less memory in comparison to the deterministic approach.**

***Keywords* – Heavy-hitters problem, Count-Min Sketch.**

I. INTRODUCTION

In many practical applications, only the most important (i.e. most frequent) values in a data stream need to be found. For example, we may need to find popular products of an online seller using the page-view information, or to find popular search queries from all searches on a search engine, or even to detect DDoS attacks using the information of incoming requests of an online service [1].

In all the examples, the goal is to find all the elements that happen more than a certain threshold. This is the so-called heavy-hitters (HH) problem, which can be stated more formally as: *Given a data stream of m elements and the parameter k, find the elements that occur at least m/k times.*

The simplest approach would be to keep a count of occurrences for all distinct elements, for instance using a dictionary. Despite always giving exact results, this solution does not scale well, since its linear memory cost is unfeasible for big data streams.

In fact, there is no exact algorithm that solves the heavy-hitters problem in one pass over the data stream while using sublinear space. Hence, there is a need for approximate algorithms, which can yield satisfactory results while respecting the space and time constraints. There have been several such algorithms proposed, such as the Frequent Count [2], and its variations like Lossy-Count [3] and Space-Saving [4]. These algorithms rely on counter-based data structures using a sublinear amount of counters. Another approach is using a sketch, which is a probabilistic data structure that allow to maintain a summary of the data while consuming substantially less space than the minimum required to store all the data. Numerous sketches have been proposed, such as the Count-Min Sketch [5] and the Count Sketch [6]. Due to the considerable theoretical and practical importance of this area, new sketches, mostly based in the Count-Min Sketch are still being introduced, as is the case of Count-Less [7], proposed in the end of last year.

In this work, a Count-Min Sketch is used to find the most frequent words in different texts. While changing its parameters (see below), the results are compared with the exact results in order to evaluate this method.

II. COUNT-MIN SKETCH

Count-Min Sketch was proposed by Cormode and Muthukrishnan [5], with the goal of providing a simple sketch data structure with a precise characterisation of the dependence on the input parameters. The basic idea behind it is using a hash function to map from items to counts. However, the hash collisions will cause different words to be mapped to the same counter, meaning that their frequency will be overestimated. The Count-Min Sketch tries to solve this problem by using multiple hash functions, with multiple arrays of counters, one for each hash function. This is based in the principle that, using more counters, the probability that a collision will occur in all of an element's counters will be necessarily lower.

More formally, the Count-Min Sketch data structure consists of an array of counters of width w and depth d, all initially set to zero. There are d different hash functions, each associated with a different row of the counter array. Once the parameters w and d are chosen the space required is fixed, as the sketch is represented by $w \times d$ counters and d hash functions.

When an update (i_t, c_t) arrives, meaning that item a_{i_t} is updated by a quantity of c_t, c_t is added to one count in each row, with the counter being determined by passing i_t to the row's hash function. This procedure is shown in Fig. 1.

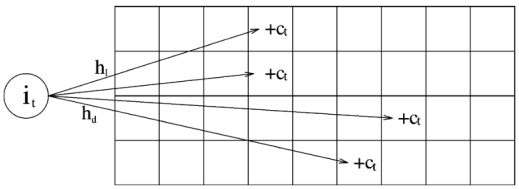

Fig. 1: Update procedure of the Count-Min Sketch. Each item i is mapped to one cell in each row. Image taken from [5].

The frequency of an element of the data stream is estimated using a query, whose procedure is similar to updates. Given an item a_{i_t}, it is passed to the d hash functions to find the cells that are mapped to that item. Then, the frequency estimation is simply the minimum value across the d counters associated with the item.

It can be shown [5] that the error in answering a query is within a factor $\varepsilon \times N$ with a probability $1 - \delta$, for $d = \lceil \log 1/\delta \rceil$ hash functions and $w = \lceil 2/\varepsilon \rceil$ columns. It should

be noted that this data structure has one-sided error, as it only returns overestimates of the true frequency counts (due to hash collisions).

Finally, the required memory space is $\mathcal{O}(1/\varepsilon \times \log 1/\delta)$ and the time per update is $\mathcal{O}(\log 1/\delta)$ [5]. Thus, the complexity of the sketch does not depend on the length of the data stream. The number of columns and rows can be chosen according to the desired accuracy and the probability of bad estimates wanted.

The Count-Min Sketch does not solve the heavy-hitters problem *per se*, but the process of finding the most frequent items using this data structure is very simple. When the number of elements of the data stream, m, is known, as in this project, the Count-Min Sketch is initialised, and all the items are registered. When the estimated frequency of an item is at least m/k, it is remembered. The recommended value of ε for this task is $\varepsilon = 1/(2k)$, because it results in the algorithm outputting every element with frequency count of at least n/k and only values with frequency count of at least $\frac{1}{2} \times n/k$. In the case of an unknown m, the process is similar but the Min-Heap data structure is used to keep the heavy-hitter candidates.

III. TEXT PREPARATION

In order to test the different instances of the Count-Min Sketch, three versions of the 1865's Jules Verne novel *From the Earth to the Moon: A Direct Route in 97 Hours, 20 Minutes* were chosen: the original French book, and both the English and Portuguese translations, obtained from the Project Gutenberg [8]-[10].

Each plain text file was downloaded and processed as follows in order to transform the plain text books into a string of lower case words, without stop-words. First, the Project Gutenberg's headers and license information were removed, in addition to the translation notes for the English and Portuguese editions. In the case of the English edition, a sequel novel that was also included in the file was removed too. Then, all punctuation characters were removed from each file. Moreover, all stop-words were removed using NLTK's [11] stop-word list extended with some custom words. Finally, all words were converted to lower case and the resulting strings were saved.

IV. RESULTS AND DISCUSSION

A. Deterministic counter

A deterministic counter was run once for each file, in order to get the exact results that would allow to evaluate the Count-Min Sketch results. These results also allowed to find some stop-words that escaped the initial list and were present among the most frequent words. The stop-word list was then updated in order to eliminate these words, the text files re-processed, and the exact counter run once again.

Before analysing the most frequent words, the total number of words was found for each book edition by summing the values of all the counters. It was found that the English edition had the fewest words (after stop-word removal), with 24,710 words, followed by the French and Portuguese versions, with 29,563 and 31,099 words, respectively. These results are in agreement with the number of letters found in the previous project for the same text files. Moreover, the number of different words in each text was also analysed. It was found that the English text has 5927 different words, the French version 8180 words and the Portuguese book contains 8965 different words.

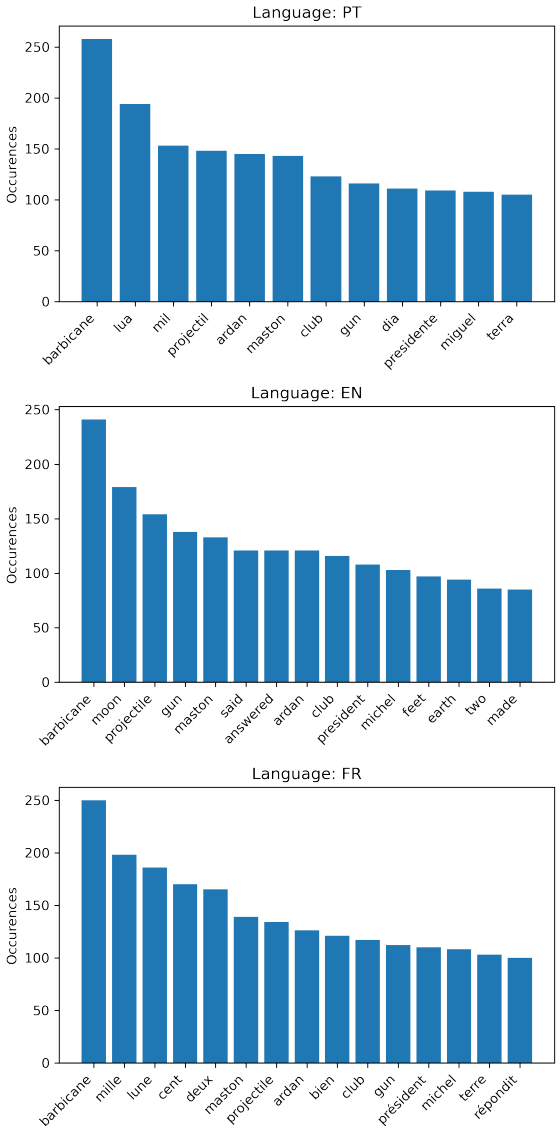

Fig. 2: Most frequent words for each language.

Then, the words that appeared at least $m/300$ times for each language were found, along with their absolute frequency. These results are shown in Fig. 2, where it can be seen that there are 12 such words for the Portuguese language, and 15 both in French and in English. Moreover, it can be observed that the most common word in all languages is 'Barbicane', which is a main character of the plot. The word 'Moon'/'Lua'/'Lune' also is one of the most frequent words in all languages, as one would expect given the book's title. Other heavy-hitter words are characters names or titles ('Ardan', 'Maston', 'President' and 'Michel'/'Miguel'), 'gun' and 'club' (Gun Club is the club where the characters meet), 'Earth'/'Terra'/'Terre', among others. The translations of 'thousand' are clear heavy-

hitters in the French and Portuguese text, but it does not appear that much in the English version. This is due to the English version translator's option of writing distances in numeral form, instead of using words.

B. Count-Min Sketch

In this section, the Count-Min Sketch was used to solve the heavy-hitters problem for $k = 300$, as done with the deterministic counter. The Count-Min Sketch was initialised with the suggested values of $\varepsilon = 1/(2 \times 300)$, which results in $m = 1200$ columns. A depth of $d = 5$ was used in this first analysis. Although it is not necessary, the implemented method was made to return the estimated frequency of each heavy-hitter word, in order to compare quantitatively with the exact counter results. The Count-Min Sketch was then run one time for each text file, and the results were compared with those of the previous section, as shown in Fig. 3.

All the estimated values are above the real counts, as expected from the sketch description. Moreover, the sketch was able to find all the heavy-hitter words, but its overestimation made it wrongly deem some words as heavy hitters for the French and English languages. For the English language, the words were 'cannon' (real count: 82, estimated: 87) and 'well', (real count: 78, estimated: 86), which were close to the 82.4 threshold to be deemed heavy-hitters. Even the order of the word frequency could be estimated with good accuracy, especially in the English and French texts, were the counts of the top words are more different from each other. In fact, the top-5 words were identified correctly for both these languages. In the Portuguese case, since word counts are closer, it was not possible to identify the top words in order. Nevertheless, in this trial it was possible to identify the group of the Portuguese top 5 words.

In order to evaluate the results quantitatively, two metrics were used: the number of words incorrectly identified as heavy-hitters despite appearing less than m/k times, and the root mean square error (RMSE) between the count estimates of the words returned by the sketch and the real count values of those words. The first metric allows to evaluate the ability of finding only the group of heavy-hitter words, with no attention to the order of the words, while the second metric gives a measure of accuracy, penalising bad frequency estimations and, consequently, estimated heavy-hitter words out of order, or with big overestimations across all words. Using these two metrics, different sketch instances can be compared: the one with lower values for both metrics is the better estimator.

For the shown results, there were 2 incorrect words for French and English, and 0 for Portuguese. In relation to the RMSE, it was higher for the Portuguese language, with 8.22, lower for French (6.86), and lowest for English (5.01). These results are expected, since we are using the same number of counters for the three texts, which have a different number of words. The higher the number of words, the worse the estimations are expected to be, and thus higher RMSEs are expected.

After obtaining and analysing these baseline results, the Count-Min Sketch parameters, m and d were varied using a grid search over depths from 1 to 10 and number of columns from $1/100$ of the recommended value to 10 times that value, using a variable step size, and sampling more values of m near 1200. These results were then analysed using the two metrics mentioned above and saved in a dataframe. To better visualise the metric values for each (m, d) pair, a heat-map of the average of the metric across the three languages was created, and as shown in Fig. 4. To produce useful visualisations, it was necessary to take the logarithm of the metric values because their range is too broad.

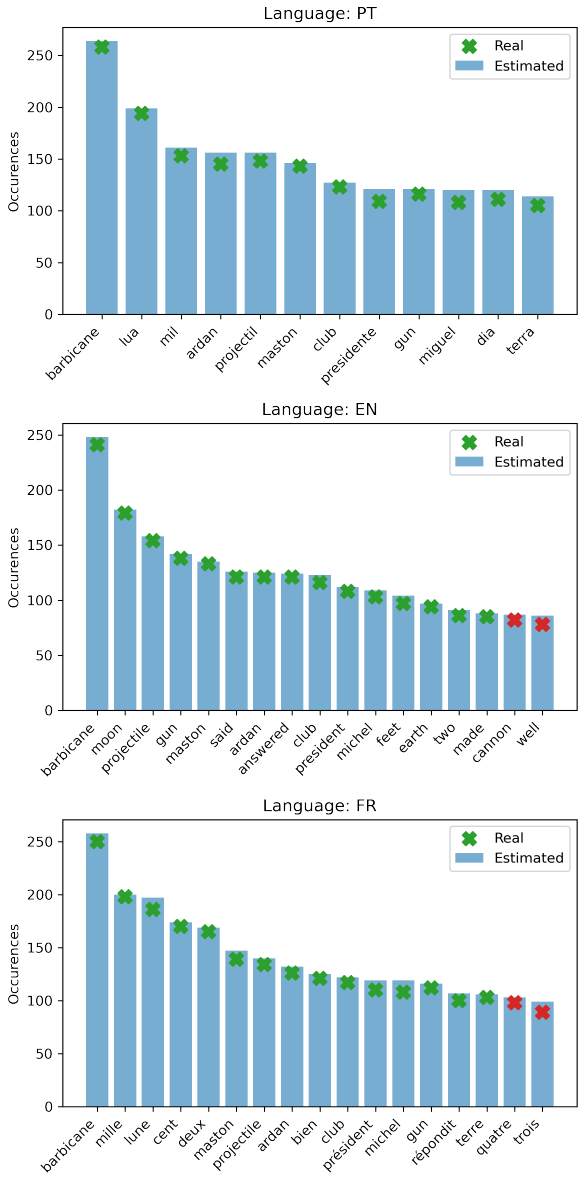

Fig. 3: Estimated heavy-hitter words for each language. Red markers mean that the respective word is not a heavy-hitter.

There is a clear gradual decrease in the two metrics both in the vertical and horizontal direction of the heatmaps. This means that, as the value of m or d is increased, the sketch returns better and better results. The decreasing trend is more clear with the increase of the number of columns, but

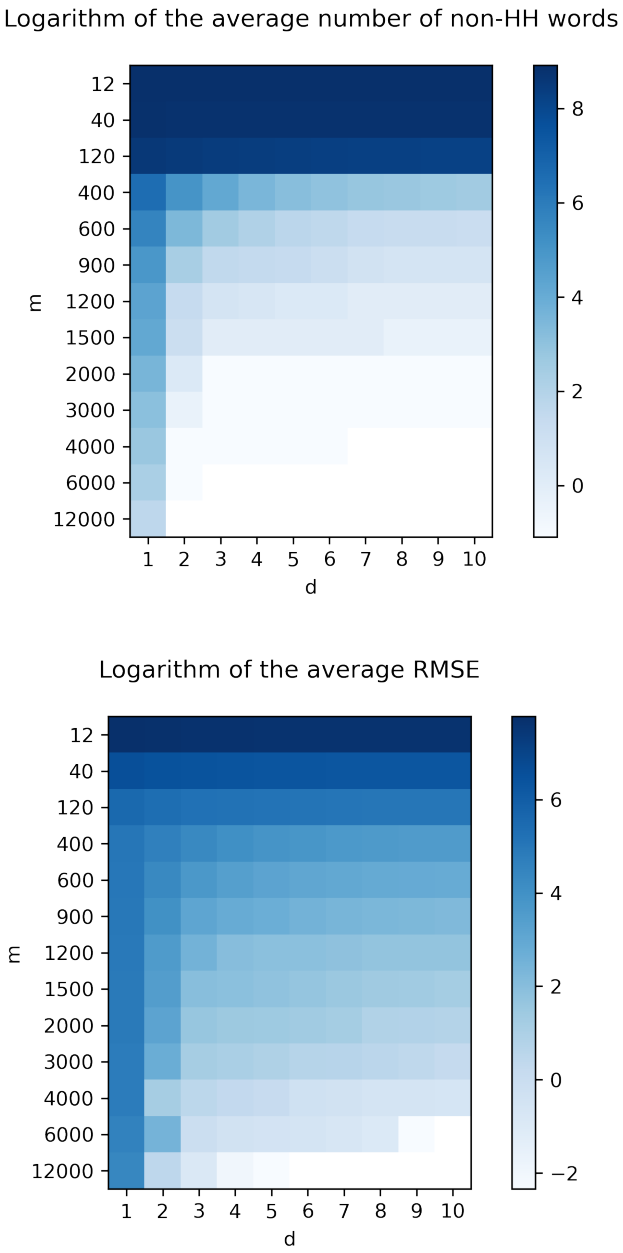

Fig. 4: Heat-maps of the two metrics for all the (m, d) pairs.

this could be explained by the fact that the parameter m ranged over a much wider interval than the parameter d. From the analysis of both heatmaps, it appears that a low (but greater than 1) number of hash functions suffices to obtain good results, given that m is no less than half of the recommended value of 1200.

Regarding the number of heavy-hitter words found in excess, for instances with 12 columns there were around 7400 misclassified words on average (almost all the words), while using 12000 columns (around a half of the total number of words, and more than the number of distinct words) allowed to only identify the true heavy-hitters. Even with 1500 columns (a 25% increase in relation to the recommended value), there was no more than one non-HH word on average, for $d > 2$. The RMSE heatmap shows a smoother decrease in comparison to the previous heatmap, due to the continuous decrease of this metric. While using 12 columns we get an average RMSE above 2000, using 12,000 columns this value was less than 1 for $d > 2$, and even 0.00, which represents perfect frequency estimation for all top words, for $d > 5$. The heatmap confirms that low frequency estimations could be made using the Count-Min Sketch, especially when $d >= 3$ and m is at least around 600.

The evolution of both metrics with one fixed parameter was also plotted, using $m = 1200$ for the fixed number of columns, as shown in Fig. 5 and $d = 5$ for the fixed number of hash functions, in Fig. 6. As shown before in the heatmap, both the metrics decrease with the increase of d. This decrease appears to be super-exponential (note that the y axis is in logarithmic units). The results for the fixed number of rows follow a similar pattern, with the super-exponential decrease very clear in the RMSE plot.

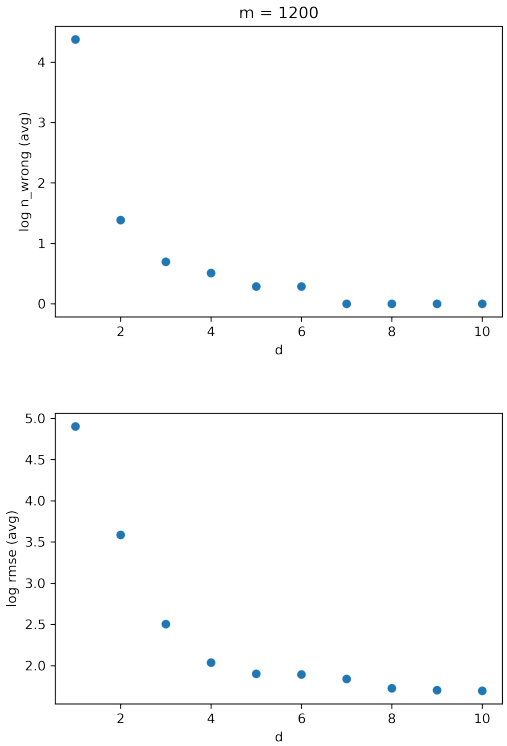

Fig. 5: Evolution of the metrics while changing the number of rows, d and keeping the number of columns constant, $m = 1200$.

Finally, it should be pointed out that the memory requirements of the sketches have not been considered so far. For instance, the baseline sketch using $m = 1200$ and $d = 5$ showed very satisfactory results, but it requires $m \times d = 6,000$ counters, which is around a fourth or a fifth of the number of words in each text. Obviously, better results can be achieved using more space, and perhaps similar results can be achieved using even less space. To follow this line of investigation, it is useful to query the metrics results dataframe for parameters which yielded an error below a certain threshold while using less than a desired number

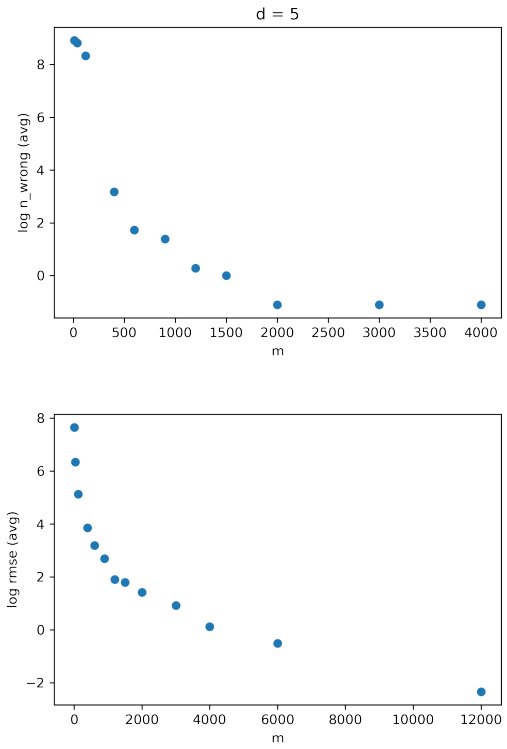

Fig. 6: Evolution of the metrics while changing the number of columns, m, and keeping the number of rows constant, $d = 5$.

of counters.

For instance, it can be found that only three parameter pairs, (m, d) obtain an RMSE lower than 20 for the English text while using a number of counters less than one eight of the total number of words. They are (600, 4), (600, 5), and (900, 3), and all shown a RMSE over 15. If we instead desire a lower RMSE, for instance below 10, while using up to one fifth of the counters (compared to one counter per word), the only options are (1200, 4) and (1500, 3), with RMSEs of 5.87 and 5.17, respectively. Incidentally, the sketch with the lower error is also the one which uses less counters. These results suggest that the baseline sketch parameters of $m = 1200$ and $d = 5$ could be changed if we desire to improve the memory requirements of the algorithm, as $(m, d) = (1500, 3)$ gives a similar RMSE as the baseline model while using 1500 counters less.

It should be remarked that the performed analysis uses the results of only one trial for each parameter pair, which can cast some doubts since we are evaluating a probabilistic algorithm. However, there were several parameter combinations sampled (130 to be exact), each over three different texts, and all the heatmaps and plots show clear trends. Thus, the analysis seems to be valid, despite obvious uncertainties in the individual metric values for each sample.

V. CONCLUSIONS

In this project, a Count-Min Sketch was successfully used to find the heavy-hitter words in different texts, with a degree of error adjustable by the sketch's parameters. The data structure used has the advantage of having perfect recall, meaning that it identifies all the heavy-hitter words, despite the possibility of also identifying some other words which are not heavy-hitters.

The sketch's effectiveness was evaluated over a range of (m, d) values, using the number of false positives and the root mean squared error between the estimated and the real absolute frequencies. It was found that with the increase in the number of sketch cells (bigger m or d), the error of the results decreased super-exponentially. Moreover, it was possible to find some pairs of parameters that achieved RMSEs lower than 20 with as few as an eight of the counters that would be needed by the deterministic approach to the heavy-hitter problem.

Thus, the Count-Min Sketch proved to be a valuable approach for finding heavy-hitter words even in the studied toy example, introducing a small estimation error in order to use much less memory. In the case of a real data stream, this data structure can be used, assisted by the Min-Heap, to reliably estimate the heavy-hitter elements.

REFERENCES

[1] Yehuda Afek, Anat Bremler-Barr, Edith Cohen, Shir Landau Feibish, and Michal Shagam, "Efficient Distinct Heavy Hitters for DNS DDoS Attack Detection", 2016.
URL: http://arxiv.org/abs/1612.02636

[2] J. Misra and David Gries, "Finding repeated elements", *Science of Computer Programming*, vol. 2, no. 2, pp. 143–152, 1982.

[3] Gurmeet Singh Manku and Rajeev Motwani, "Approximate frequency counts over data streams", in *Proceedings of the 28th International Conference on Very Large Data Bases*. 2002, VLDB '02, p. 346–357, VLDB Endowment.

[4] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi, "Efficient computation of frequent and top-k elements in data streams", in *Database Theory - ICDT 2005*, Thomas Eiter and Leonid Libkin, Eds., Berlin, Heidelberg, 2005, pp. 398–412, Springer Berlin Heidelberg.

[5] Graham Cormode and S. Muthukrishnan, "An improved data stream summary: The count-min sketch and its applications", *Journal of Algorithms*, vol. 55, no. 1, pp. 58–75, 2005.

[6] Moses Charikar, Kevin Chen, and Martin Farach-Colton, "Finding frequent items in data streams", *Theoretical Computer Science*, vol. 312, no. 1, pp. 3–15, 2004.

[7] SunYoung Kim, Changhun Jung, RhongHo Jang, David Mohaisen, and DaeHun Nyang, "Count-Less: A Counting Sketch for the Data Plane of High Speed Switches", pp. 1–16, 2021.
URL: http://arxiv.org/abs/2111.02759

[8] "Project Gutenberg: The Moon-Voyage by Jules Verne", https://www.gutenberg.org/ebooks/12901, Accessed: 28/12/2021.

[9] "Project Gutenberg: De la Terre à la Lune by Jules Verne", https://www.gutenberg.org/ebooks/799, Accessed: 28/12/2021.

[10] "Project Gutenberg: Da terra à lua, viagem directa em 97 horas e 20 minutos by Jules Verne", https://www.gutenberg.org/ebooks/28341, Accessed: 28/12/2021.

[11] Steven Bird, Ewan Klein, and Edward Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*, O'Reilly Media, Inc., 2009.