Supplement for "Indexing Strings with Utilities"

Giulia Bernardini[1], Huiping Chen[2], Alessio Conte[3], Roberto Grossi[3], Veronica Guerrini[3],
Grigorios Loukides[4], Nadia Pisanti[3], and Solon P. Pissis[5]

[1]University of Trieste, Trieste, Italy [2]University of Birmingham, Birmingham, UK [3]University of Pisa, Pisa, Italy
[4]King's College London, London, UK [5]CWI & Vrije Universiteit, Amsterdam, The Netherlands

I. Additional Material for Section VII

A. *SubstringHK*

In this section, we present an example of an input for which SubstringHK can fail. Consider the string $S = (\mathtt{AB})^{n/2}$, namely $\mathtt{AB}$ repeated $n/2$ times, where $n/2 \geq K > 4$, K is even, and $|\Sigma| = 2$ (other choices are possible). By construction, its top-K most frequent substrings are $\mathtt{A}, \mathtt{B}, \mathtt{AB}, \mathtt{BA}, \ldots, (\mathtt{AB})^{K/4}, (\mathtt{BA})^{K/4}$; however, almost half of the output (i.e., $\mathtt{BA}$, $\mathtt{BAB}$, $\mathtt{BABA}, \ldots, (\mathtt{BA})^{K/4}$) is not reported, as we report next.

At the initial position $i = 0$, substrings $S[0] = \mathtt{A}$, $S[0\,..\,1] = \mathtt{AB}$, up to $S[0\,..\,K] = (\mathtt{AB})^{K/2}\mathtt{A}$ are considered and all inserted in *ssummary* but the last one, as it has the same frequency as the previous K ones. At the next position $i = 1$, $S[1] = \mathtt{B}$ is not inserted into *ssummary* as its frequency does not exceed those in *ssummary*. Next, the first K substrings starting at $i = 2$ (i.e., $S[2] = \mathtt{A}$, $S[2\,..\,3] = \mathtt{AB}$, up to $S[2\,..\,2 + K - 1] = (\mathtt{AB})^{K/2}$) are already in *ssummary* and their frequencies increase; again, $(\mathtt{AB})^{K/2}\mathtt{A}$ is not inserted. In general, the first K substrings starting at even positions $i = 0, 2, 4, \ldots$, are in *ssummary*; and no substring starting at odd positions $i = 1, 3, 5, \ldots$ is ever inserted in *ssummary*. The reason is that the frequencies of $\mathtt{B}$ or $(\mathtt{AB})^{K/2}\mathtt{A}$ are never larger than those of the substrings in *ssummary*. In the last $K-2$ positions of S, a small change occurs: $\mathtt{B}$ is now able to enter *ssummary* (but not $\mathtt{BA}$), but there are too few positions left to allow any substring starting with $\mathtt{B}$ of length greater than one to enter.

Intuitively, this is the reason why SubstringHK fails to detect long frequent substrings, as we prove also experimentally in Section IX of the main manuscript.

B. *Top-K Trie*

In this section, we present an example of a string for which Top-K Trie fails to report half of the output. Consider the string $S = \mathtt{A}^{n/2}\mathtt{B}^{n/2}$ with $n \geq 2K(K+1)$, for which the top-K frequent substrings are $\mathtt{A}$, $\mathtt{B}$, $\mathtt{AA}$, $\mathtt{BB}$, $\ldots$, $\mathtt{A}^{K/2}$, $\mathtt{B}^{K/2}$. The only substrings Top-K Trie reports are $\mathtt{B}$, $\mathtt{BB}$, $\ldots$, $\mathtt{B}^{K}$, because of the reasons we illustrate next.

At the beginning, the trie comprises only the root. At each position i of S, the algorithm takes $S[i]$ and a node v of the trie, and checks if v has a child u with an edge labeled $S[i]$. If so, the counter of u is incremented by one and the node considered for the next position is u. Otherwise, we have two cases according to the trie size (i.e., the number of its edges) and the children of v: *(i)* If the trie size is less than K, then a new child u is added to v with edge label $S[i]$ and its counter is set to 1. *(ii)* If the trie size is K, then all the counters of the trie are decreased by one and the nodes having counters equal to 0 are removed. In both cases *(i)*-*(ii)*, v is reset to be the root for the next position. The top-K most frequent substrings that are maintained are outputted via a traversal of the trie.

The number of considered substrings in S is at most $|S| = n$. Top-K Trie uses $\mathcal{O}(K)$ space and processes each letter of S in constant amortized time, thus reporting the approximate top-K substrings in $\mathcal{O}(n + K)$ time. As SubstringHK, this strategy fails to identify frequent long substrings in practice (see Section IX of the main manuscript). This is because the frequency of substrings stored in the trie is underestimated, which results in longer substrings having a greater probability of being removed from the trie.