Query-time false positives k-mer filtration and MultiFilters

Lucas Robidou and Pierre Peterlongo

Inria, Rennes

February 11, 2021

Bloom Filters

About Bloom filters

A Bloom filter is a data structure designed to tell rapidly and memory-efficiently whether an element W is present in a set S.

- If W is in S, the the query return "true".
- If W is **not** in S, the query **may still return "true" with a probability of** ϵ.

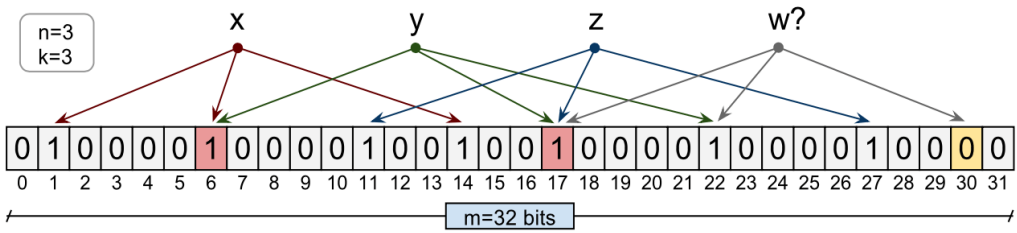


Figure: A Bloom filter (each element have 3 hash functions)

source: https://www.abhishek-tiwari.com/bloom-filters-is-element-x-in-set-s/

Some characteristics:

- The possible answers are only booleans ("yes" or "no")
- If n represents the number of elements stored, a Bloom filter requires at least $-\frac{n \ln(\epsilon)}{(\ln(2))^2}$ bits.
  (the memory grows in $O(n)$)

How to identify false positive results ?

Context

Compute similarity of an index with a queried sequence:

- extract every k-mer
- query each k-mer
- compute the similarity

Exemple of a query

indexed sequence: 

query: 

Figure: Execution of a query on a perfect filter

Exemple of a query

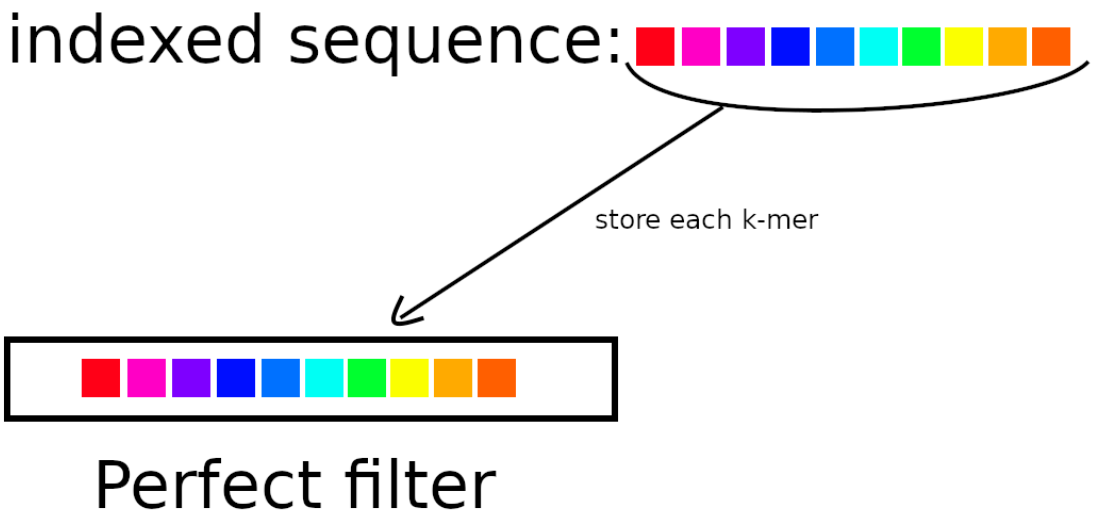


Figure: Execution of a query on a perfect filter

Exemple of a query

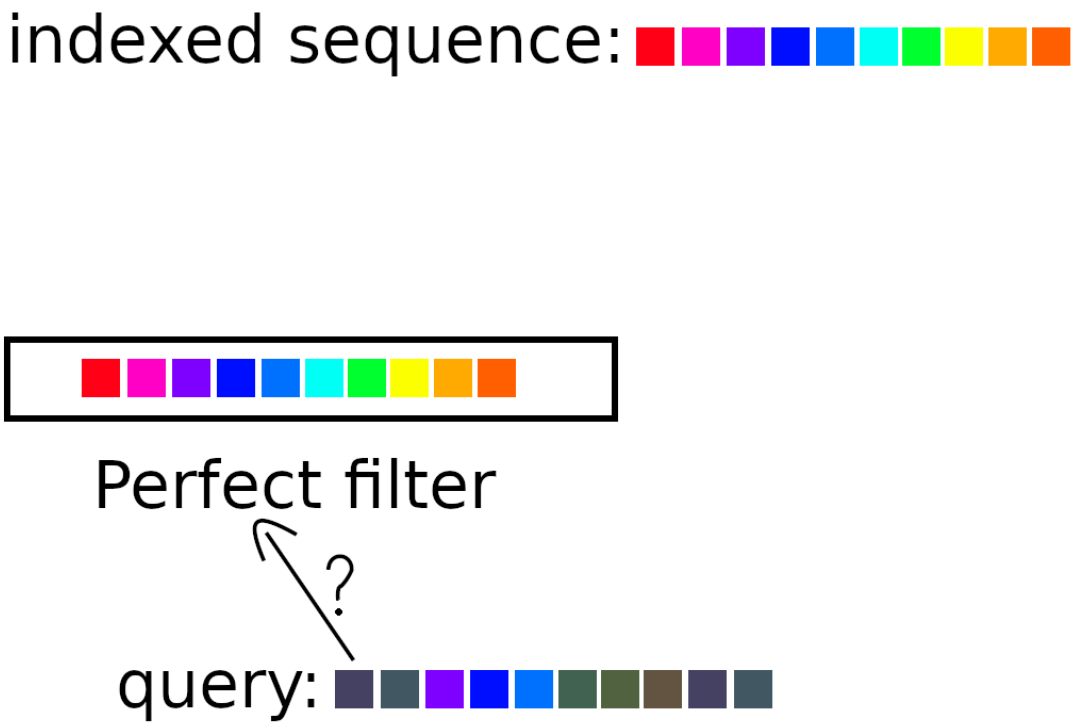


Figure: Execution of a query on a perfect filter

Exemple of a query

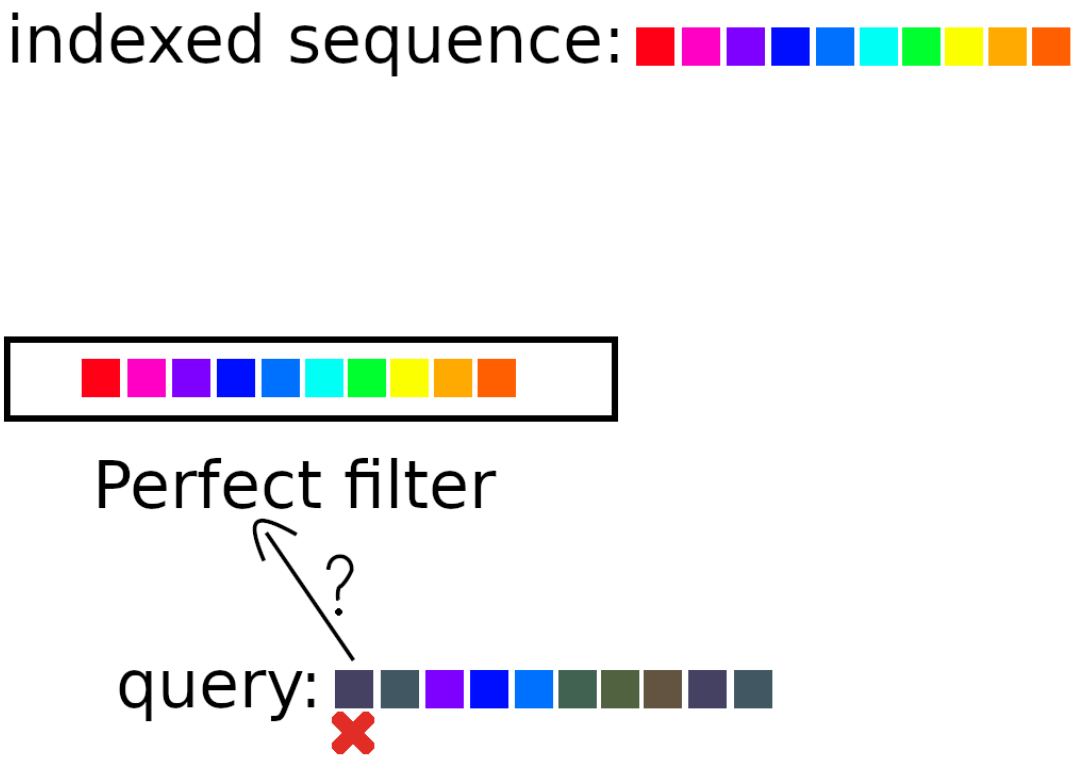


Figure: Execution of a query on a perfect filter

Exemple of a query

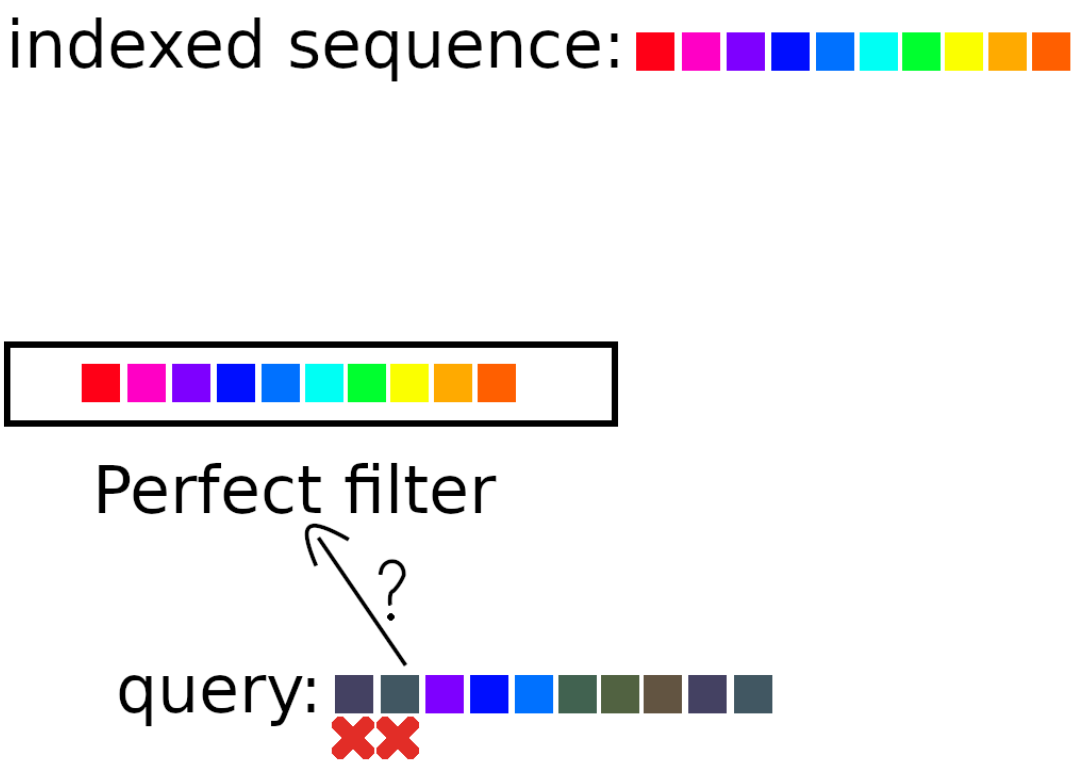


Figure: Execution of a query on a perfect filter

Exemple of a query

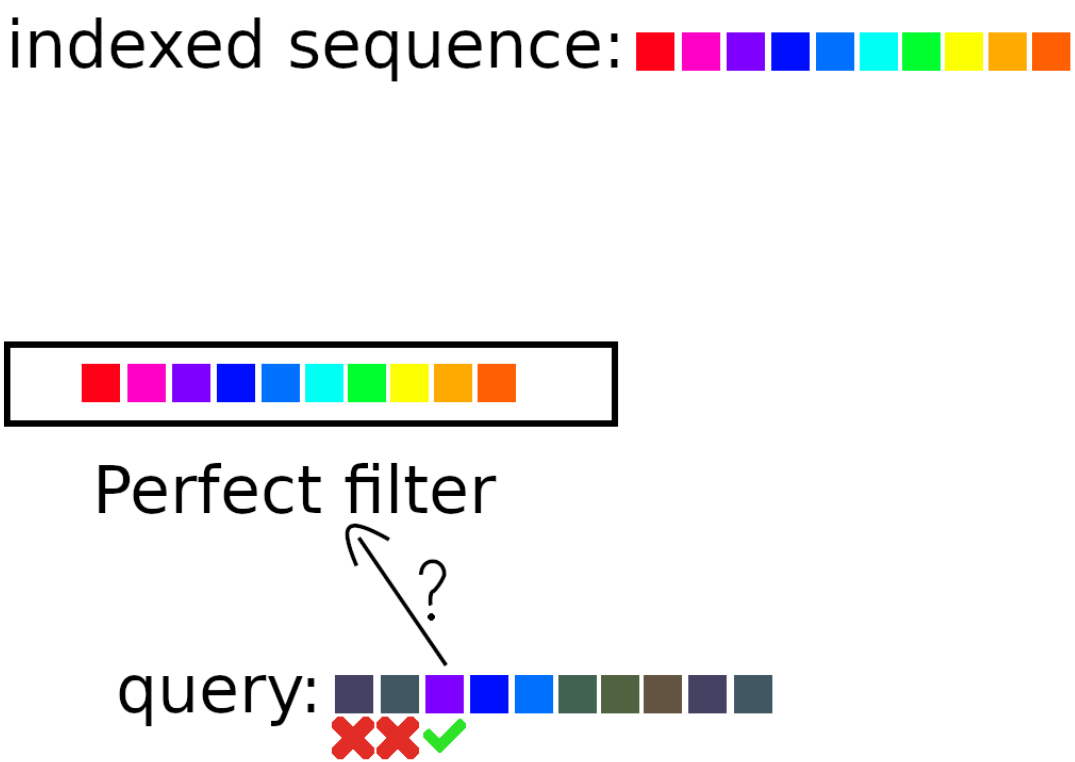


Figure: Execution of a query on a perfect filter

Exemple of a query

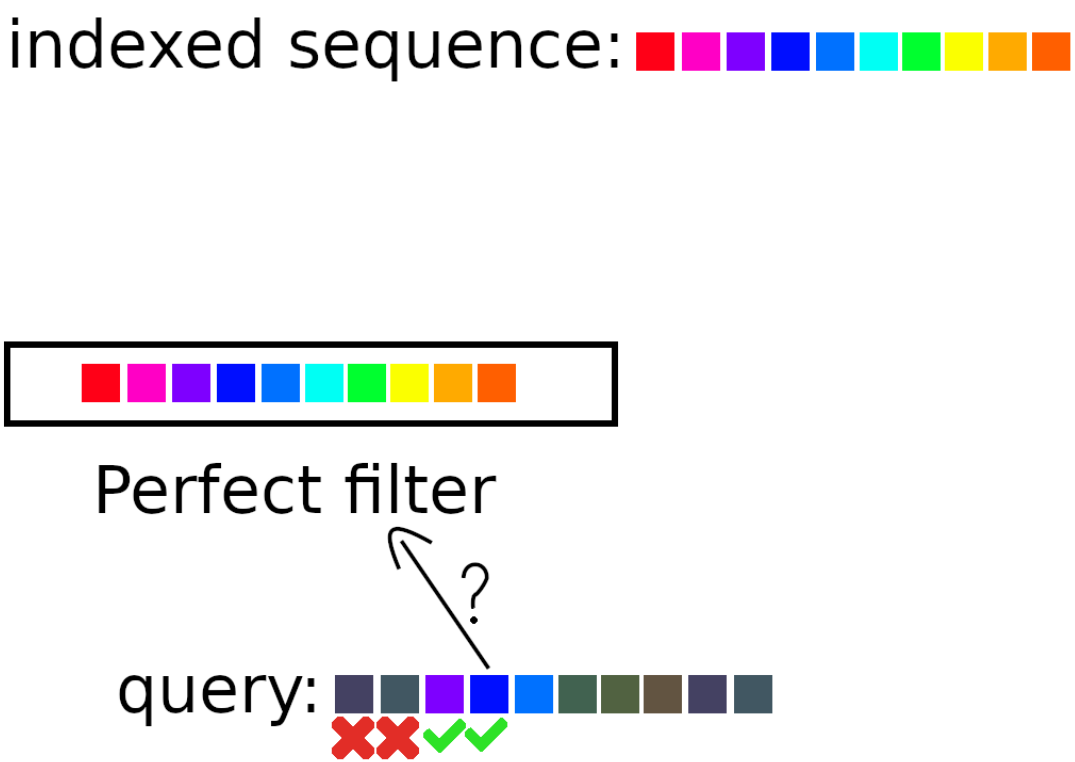


Figure: Execution of a query on a perfect filter

Exemple of a query

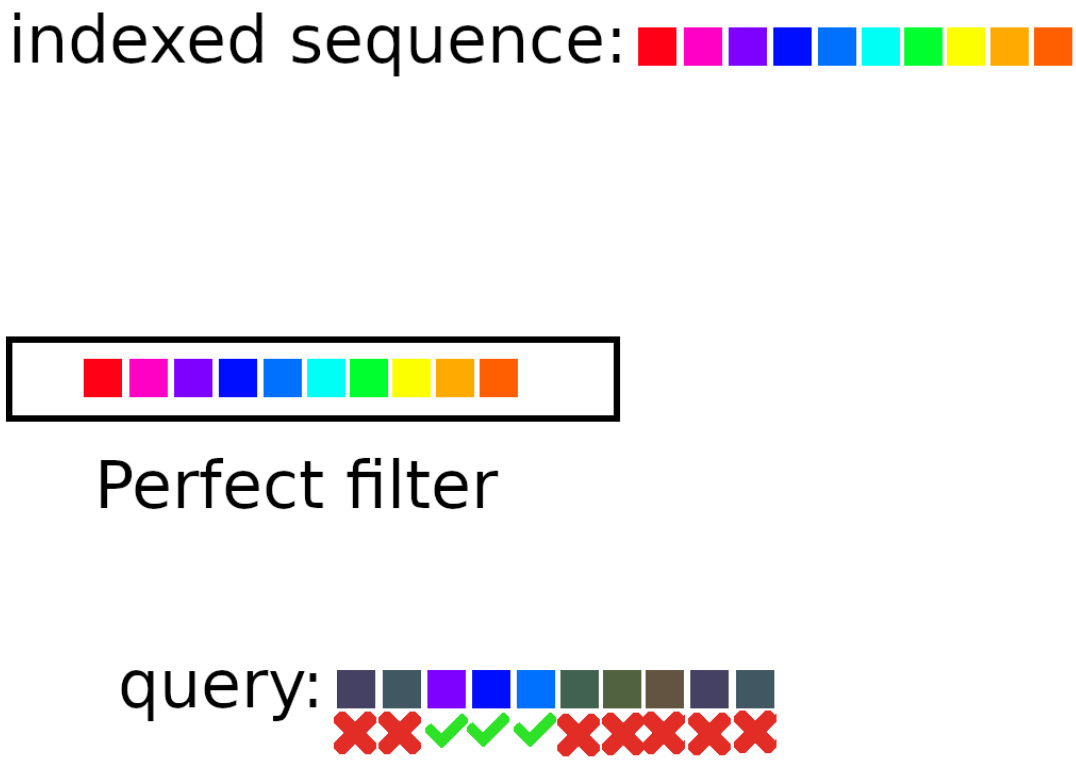
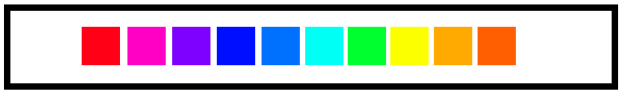


Figure: Execution of a query on a perfect filter

Exemple of a query

indexed sequence: 



Bloom filter

query: 

Figure: Execution of a query on a Bloom filter

Time for some definitions

- Positive k-mer: k-mer that is found in a Bloom filter (might be a FP)
- Neighbours of a k-mer: k-mers located on the "left" and on the "right" of a given k-mer
- Z-positive stretch: stretch of Z positives neighbours
- Isolated k-mer: k-mer part of a 1-positive stretch (no positive neighbours)

Exemple of a query with only one isolated kmer

Figure: Execution of a query on a Bloom filter

Low probability to have one isolated shared kmer between two similar sequences.

Main idea

Compute similarity of an index with a queried sequence:

- **choose Z**
- extract every k-mer
- query each k-mer
- **if a kmer is not part of a $>$Z positive stretch, consider it as a negative**
- compute the similarity

Preliminary results

Using a Bloom filter:

- $\epsilon = 4\%$
- $k = 32$

Indexing whole gnenomes of:

- 2 E. Coli
- 1 Listeria phage
- 1 Penicillium chrysogenum

Querying another E. Coli with $Z = 6$

We were able to :

- decrease the false positive rate from 4% to $\approx 0.005\%$ $(FPR = \frac{FP}{FP+TN})$
- **however**: we had a false negative rate of $\approx 3.65\%$ $(FNR = \frac{FN}{FN+TP})$

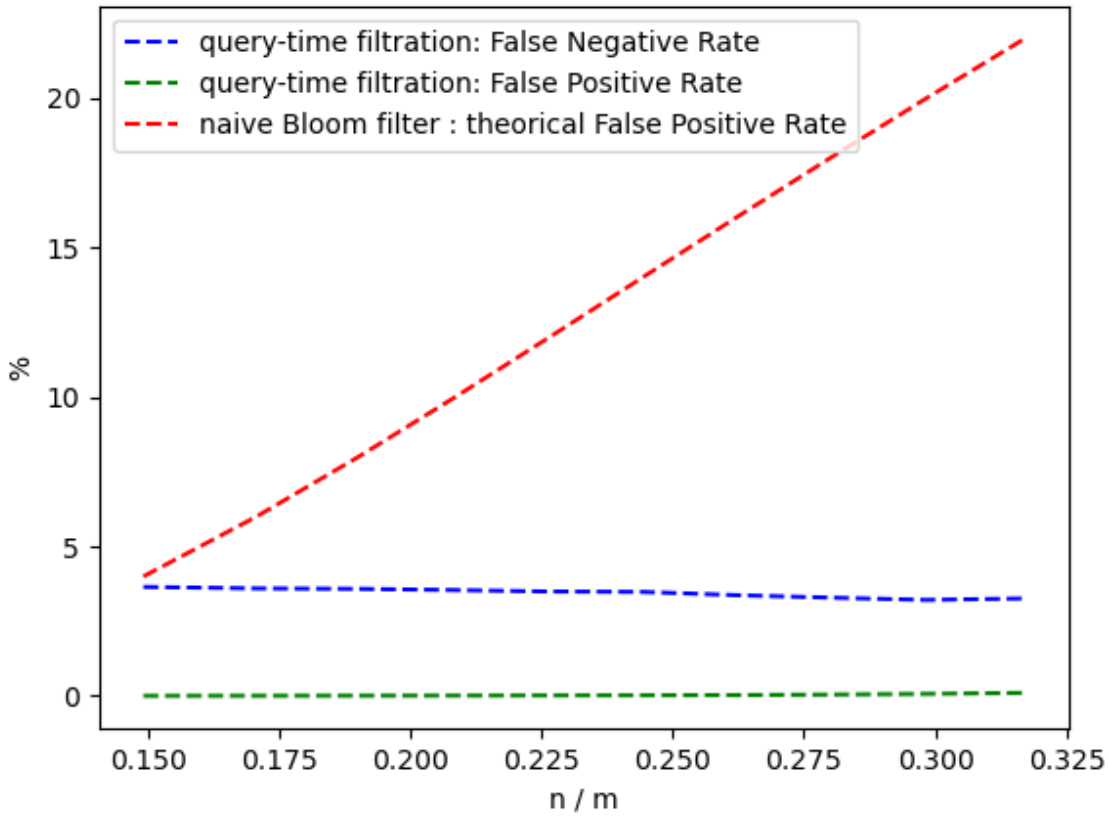


Figure: *FPR* anf *FNR* of (Bloom filter + query-time filtration) vs "naive" Bloom filter

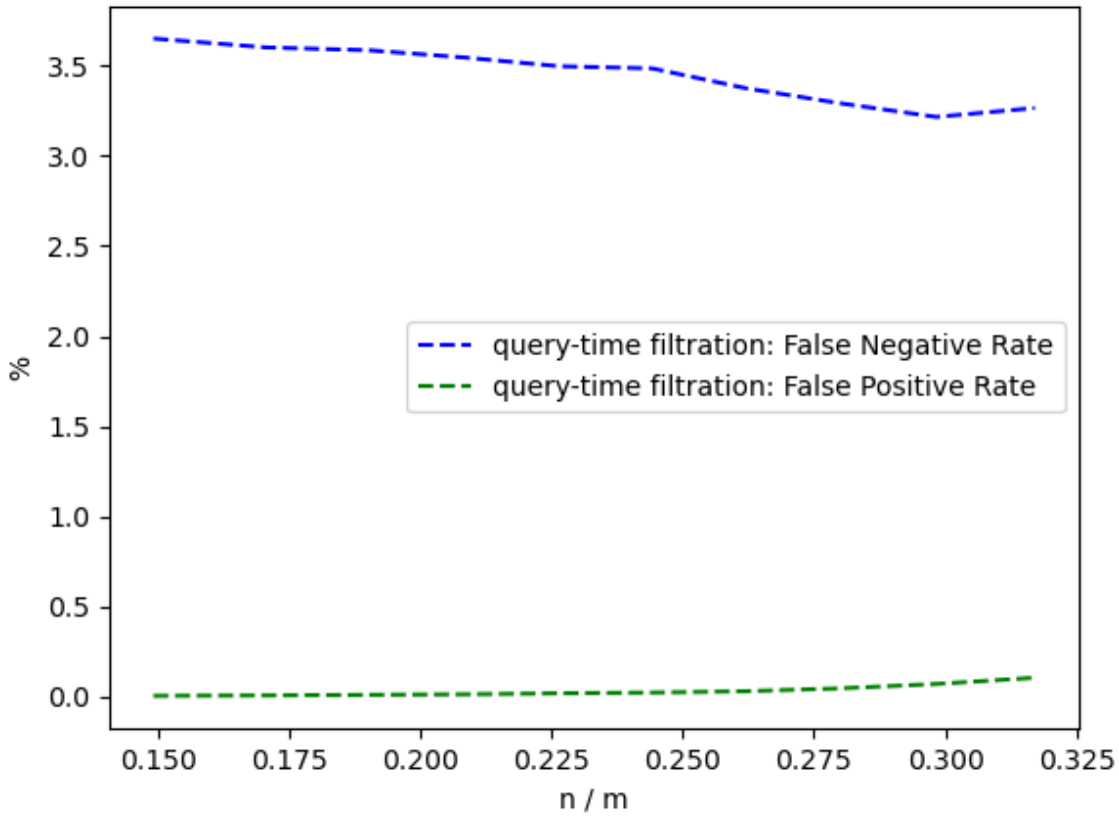


Figure: *FPR* anf *FNR* of Bloom filter + query-time filtration

Multifilters

Main idea of Multifilters

Remember : "The possible answers are only booleans ("yes" or "no")"
– >Storing kmer abundance: one bloom filter per abundance (or range of abundances)

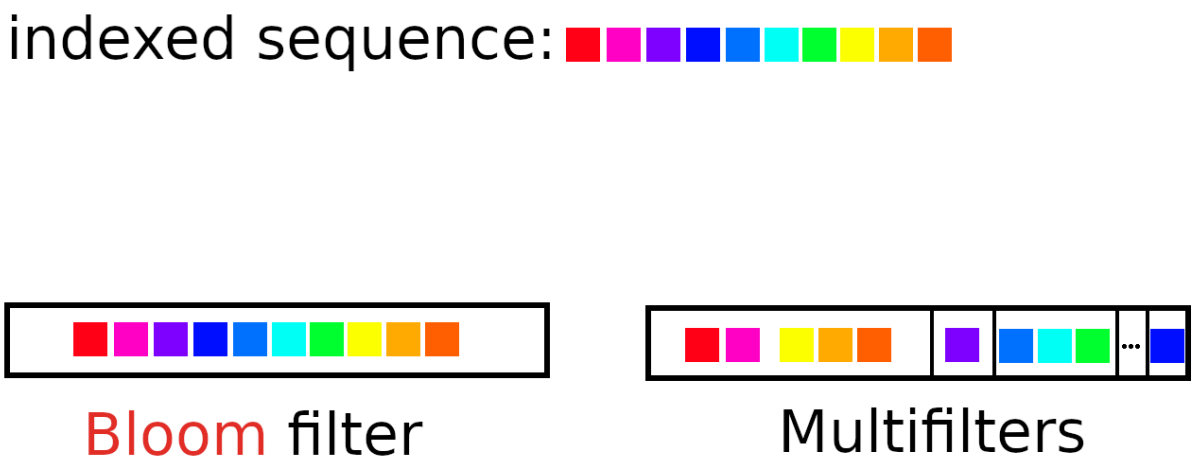
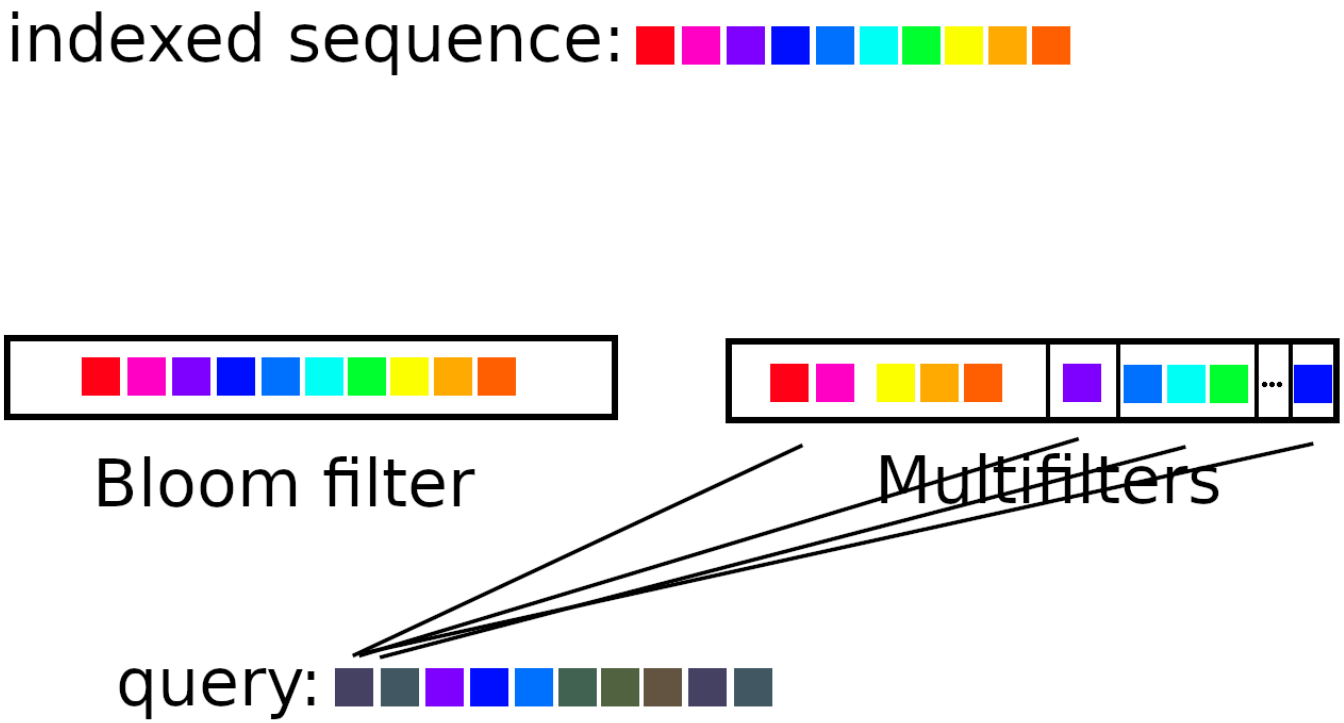
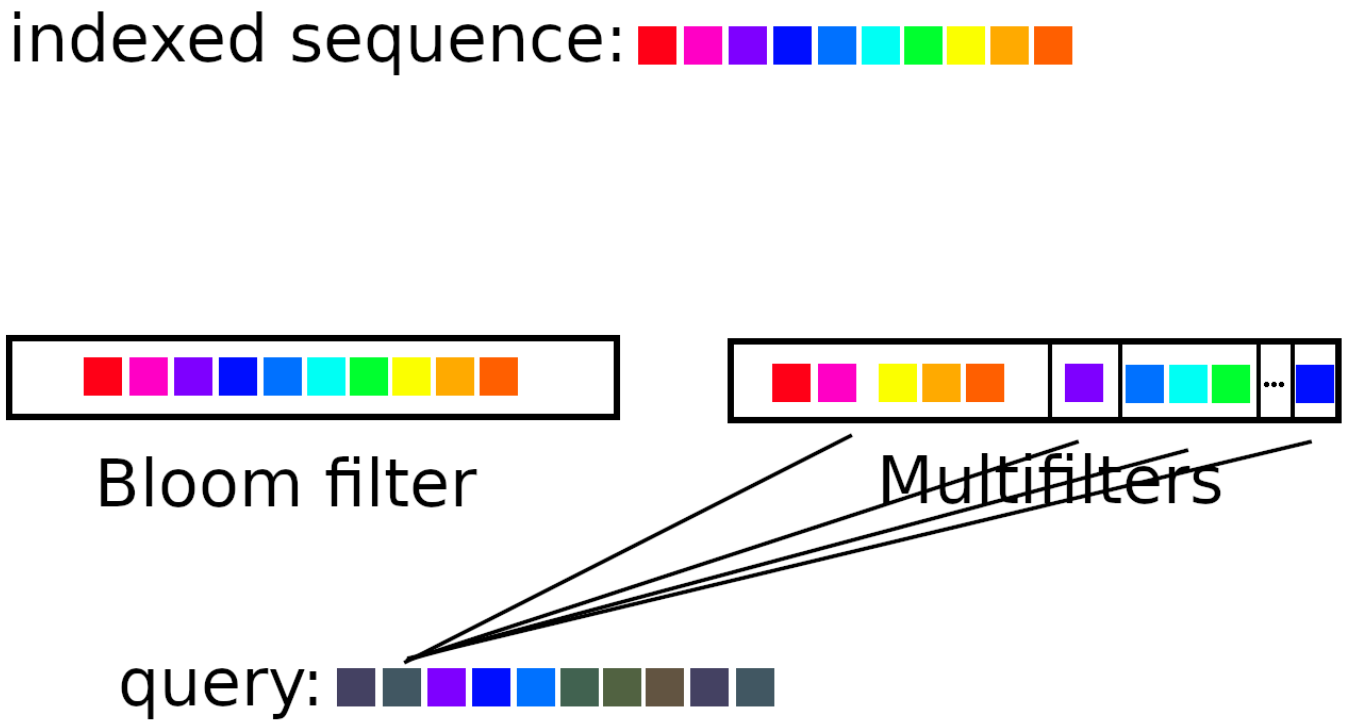


Figure: Main idea of Multifilters

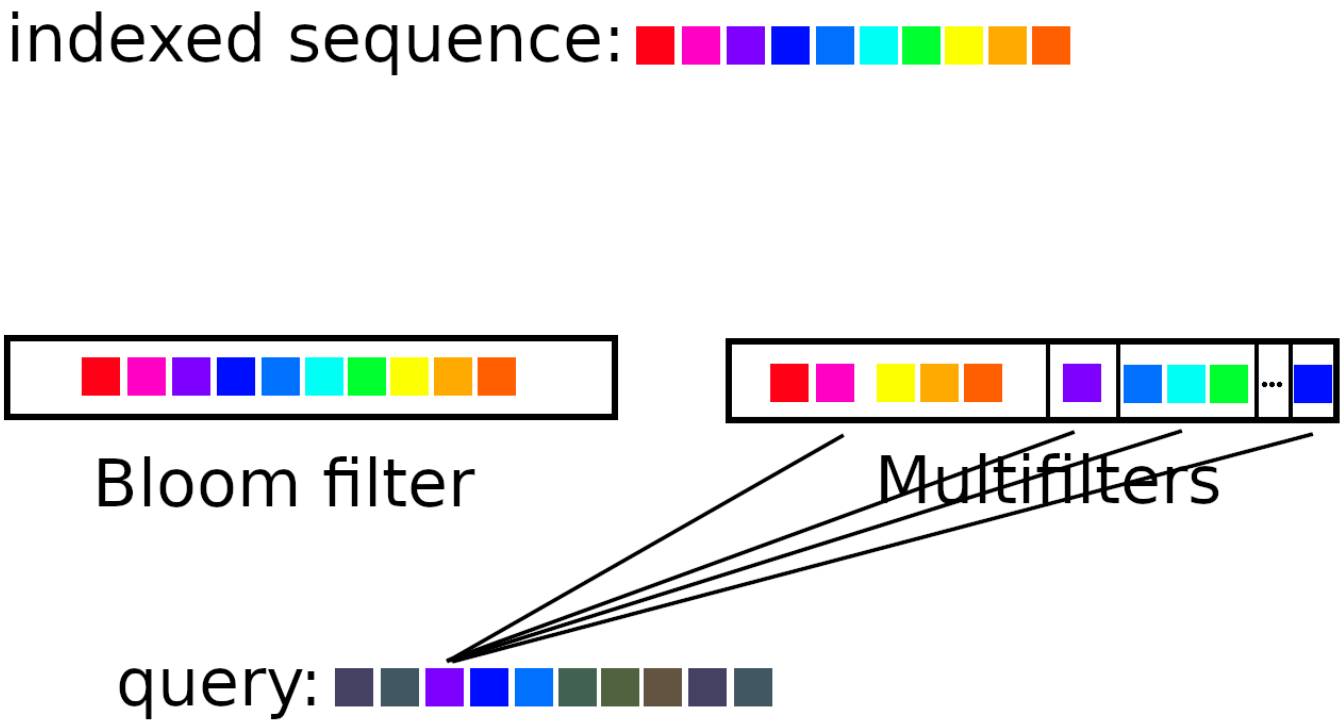
Usage of Multifilters



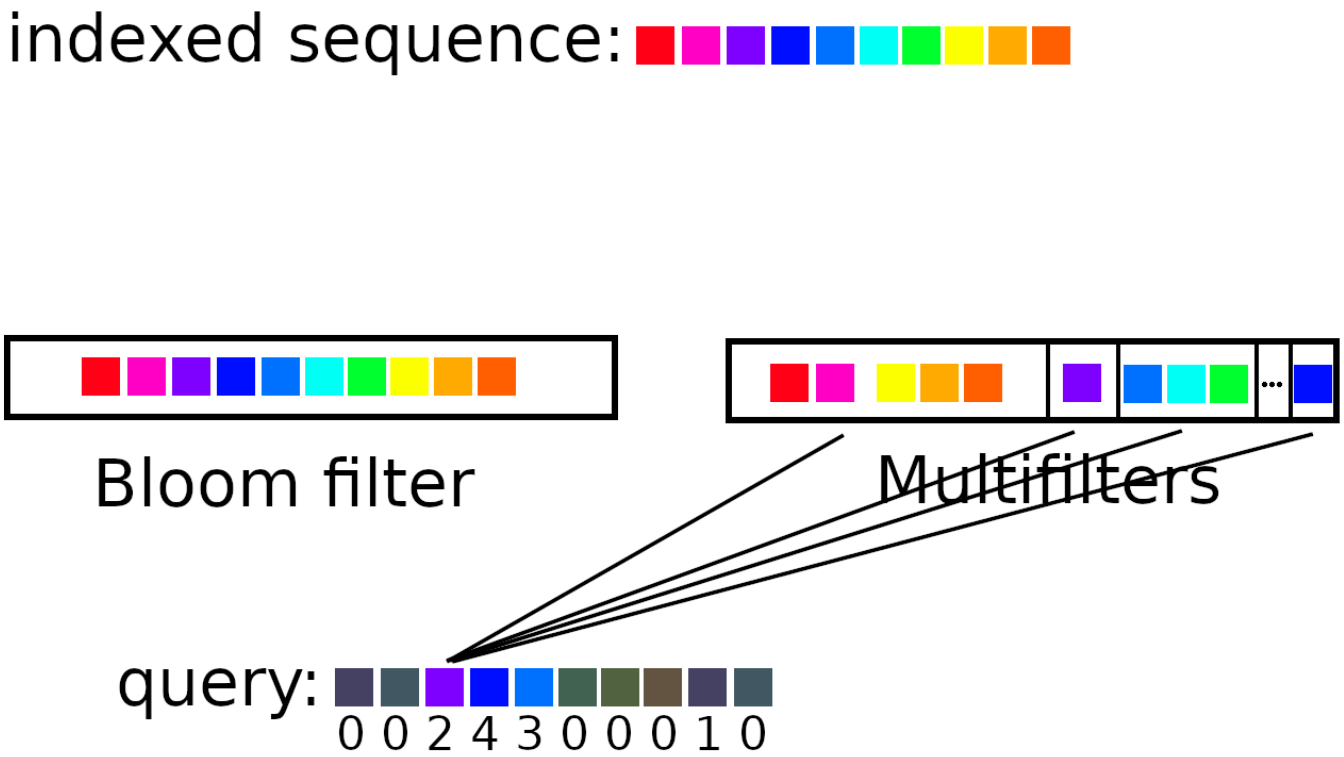
Usage of Multifilters



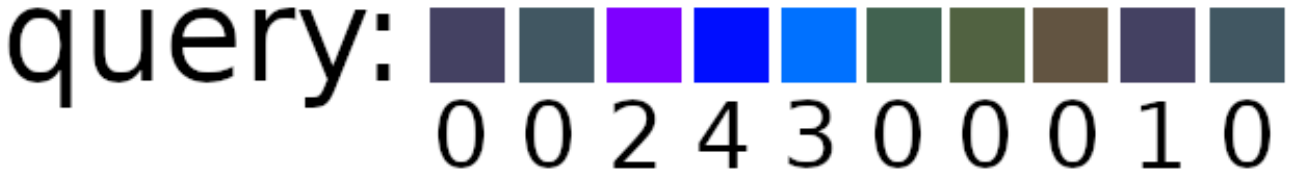
Usage of Multifilters



Usage of Multifilters



Usage of Multifilters

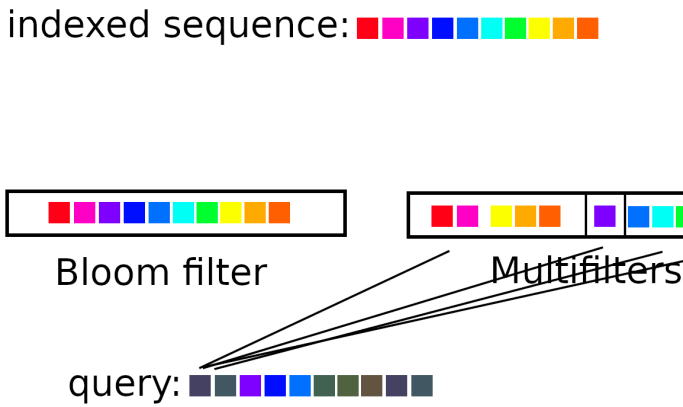


Main problem Multifilters

More queries : more false positives !

One possible solution is to lower ϵ for each filter, but each "sub filter" ends up taking as much space as the original Bloom filter...

If only we had a solution to handle so many false positives !



Applying query-time filtration

Compute similarity of an index with a queried sequence:

- extract every k-mer
- query each k-mer **on each sub filter**
- if a kmer is not part of a >Z positive stretch **on a sub filter**, consider it as a negative
- **else, select the filter for which the number of positives neighbours is maximized**
- compute the similarity

Exemple of an execution

This way, we are able to get rid of most false positives:

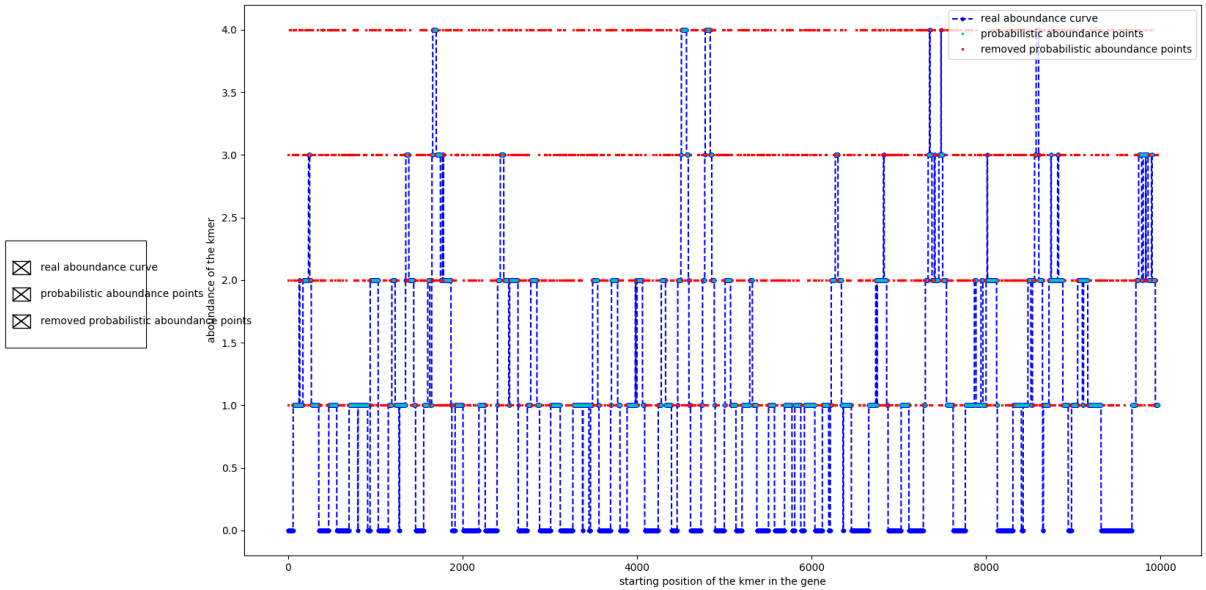


Figure: Execution of a query on a Multifilter with query time filtration

Exemple of an execution

This way, we are able to get rid of most false positives:

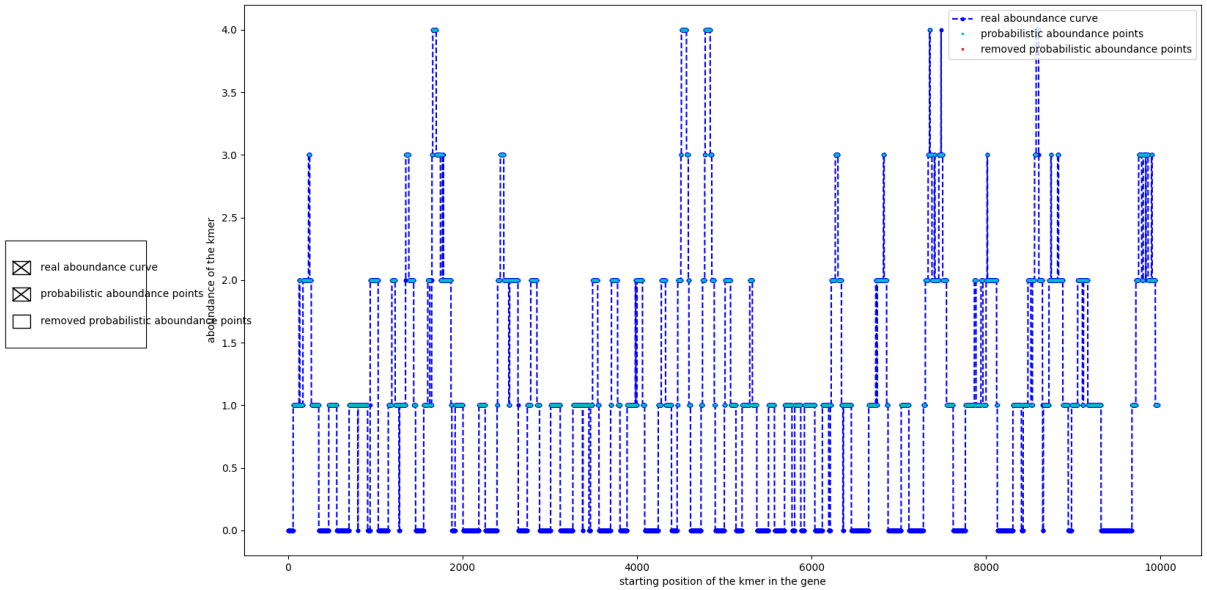


Figure: Execution of a query on a Multifilter with query time filtration

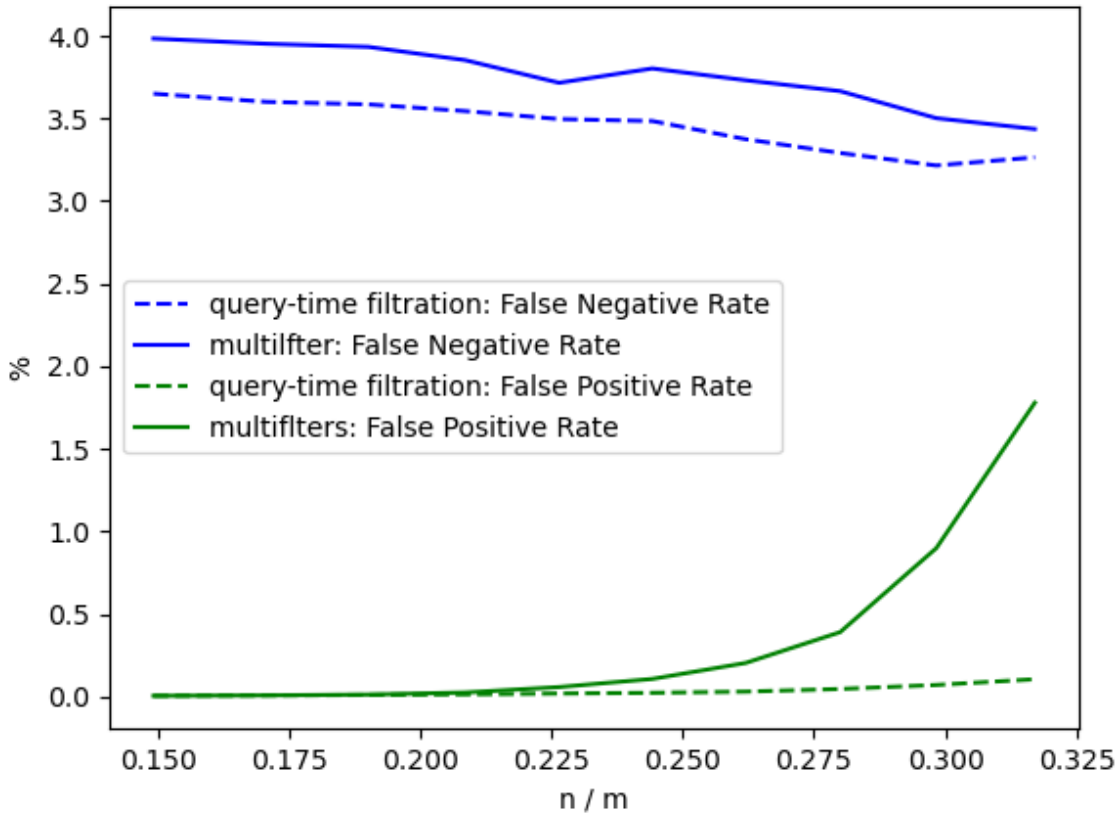


Figure: *FPR* anf *FNR* of (Bloom filter + query-time filtration) vs (Multifilters + query-time filtration)

Take home message

By using one Bloom filter per abundance, we can:

- index kmers along with their abundance
- without needing more memory
- with about 800 times less false positives

The price to pay is about 4% of false negatives rate, which is 4% more than Bloom filters.