UNIVERSITY OF ZAGREB, FACULTY OF ELECTRONICS AND COMPUTING

Computing optimal tresholds for q-gram filters

Stipe Kuman, Dino Radaković, Leon Rotim

January 13, 2016

1 INTRODUCTION INTO THE PROBLEM

In this project we have analyzed and recreated a paper on the topic of finding optimal thresholds for q-gram filters. Q-gram filters are used for matching substrings of length q in a text with substrings in a given pattern. In addition, the paper we are recreating uses gapped q-grams, which allow for discontinuous text substrings. Gapped q-grams were shown to be more efficient than contiguous q-grams in the right conditions. However, computing an optimal threshold for those filters is also more difficult. Our project was done in C++ and can run on the bio-linux platform. In the next section we will describe the process we used in our computation. Before that, we will describe some of the main principles our algorithm is based on.

1.1 PROBLEM DESCRIPTION

First off, we will describe the notation used in this assignment. T is the text file, P is the given pattern and S stands for all the substrings in T that match the given pattern P within a Hamming distance of k (number of non-matching characters). The length of the q-gram filter is q and the threshold is represented by t. Finally, m is the length of P and S.
The goal of the q-gram filter is to reduce the number of potential matches that need to be compared to the given pattern P. The threshold variable t defines the minimum number of q-gram matches between the pattern P and a substring so that the substring would be considered as a potential match. A low value of t will result in too many potential matches, while a too high value may overlook some potential matches which would make the filter lossy. Only filters that label all the matches as potential matches will be considered in the assignment.
As stated previously, this assignment will focus on gapped q-grams. For contiguous q-grams an optimal threshold can be computed with a simple mathematical formula, which is not true

for gapped q-gram filters. In gapped q-gram filters the problem is much more complex because a closed form formula has not been found. Because of that we have used the dynamic algorithm found in the assignment paper.

1.2 PRUNING METHOD

Even with this reasonably fast dynamic programming algorithm computing all possible gapped Q-grams with positive threshold for parameters $m = 50$, $k = \{4,5\}$ is still a challenging task. Using brute force method, calculating threshold for all shapes in that have $m = 50$ and $k = \{4,5\}$ and leaving only those for which positive threshold is computed, would leave us with search space of $2^{50} (\approx 10^{15})$ thresholds to compute. Since that kind of search space is clearly unfeasible we use fallowing lemma as stated in [**?**] to reduce search space.

Lemma 1.1. *If* $Q' \subseteq Q$ *then* $threshold_{Q'}(m,k) \geq threshold_Q(m,k)$

Using lemma 1.1 we propose simple pruning technique which we used while computing all Q-shapes with positive thresholds.
We introduce parameter MS as a maximal span size of all Q-grams that will considered. Obviously $MS \leq m$. This parameter is important because its value drastically changes search space.
Let $Set_{current}$ be a set which contains all Q-grams with size q, which respective thresholds had been computed. Next we define two empty sets, Set_{next} and $Set_{forbidden}$, which will contain Q-grams of size $q+1$. After running pseudocode 1 Set_{next} will contain all Q-grams of size $q+1$ that can not be eliminated with lemma 1.1.

Algorithm 1 Generating candidate set of Q-grams which may have positive threshold

function generateNextCancidateSet ($Set_{current}, MS$)

Input : $Set_{current}$ - a set of Q-grams of size q, MS - maximal value of span for all considered Q-grams

Output: Set_{next} - set of Q-grams of size q+1 that can't be eliminated using lemma 1.1

```
Set_next ← ∅
Set_forbidden ← ∅
for each Q-gram Q_i ∈ Set_current do
    for i ∈ {2,...,MS} \ Q_i do
        Q'_i = Q_i ∪ i
        if threshold(Q_i) == 0 then
            Set_next ← Set_next \ Q'_i
            Set_forbidden ← Set_forbidden ∪ Q'_i
        else if Q'_i ∉ Set_forbidden then
            Set_next ← Set_next ∪ Q'_i
        end if
    end for
end for
```

return Set_{next}

Using procedure 1 we can compute thresholds for all Q-grams with positive thresholds with $m = 50$ and $k = \{4,5\}$ as presented in following algorithm 2.

Algorithm 2 Pruning technique for computing all Q-grams with positive threshold

function findAllQgramsWithPositiveThreshold (m, k, MS)

Input: m - size of a pattern, k - number of miss-matches, MS - maximal value of span for all considered Q-grams

```
Set_current ← {Q_start}, where Q_start = {0}
while Set_current ≠ ∅ do
  for each Q_i ∈ Set_current do
    if threshold(Q_i) > 0 then
      add pair (Q_i, threshold(Q_i)) to results
    end if
    Set_current ← generateNextCancidateSet(Set_current, MS)
  end for
```

Using this simple pruning method we drastically reduced search space by several orders of magnitude. Leaving about 10^8 thresholds to compute in order to acquire all Q-shapes with positive thresholds with $m = 50$ and $k = \{4,5\}$. Results obtained with this method are showed in tables 1.1 and 1.2. We note that tables do not contain threshold values for all possible shapes. Even with this effective pruning technique number of thresholds that has to be computed is still large. Due to the time limit and shortage of computational power results are obtained with parameter $MS = 30$ (maximal span of considered Q-grams was 30). Experiments were also ran with $MS = 50$, however more time was needed to compute thresholds for Q-grams with $q > 6$.

1.3 Results

In this section we present results obtained with for Q-grams with $m = 50$ and $k = \{4,5\}$. Maximum values of thresholds in regard to Q-gram size (q) and span are showed in tables 1.1 and 1.2.

Span	q												
	2	3	4	5	6	7	8	9	10	11	12	13	14

Table 1.1: Table showing maximal thresholds for Q-gram shapes with regard to size(q) and span of the shape for $m = 50$ and $k = 4$

Span	q										
	2	3	4	5	6	7	8	9	10	11	12
2	39	-	-	-	-	-	-	-	-	-	-
3	38	33	-	-	-	-	-	-	-	-	-
4	37	32	27	-	-	-	-	-	-	-	-
5	36	31	26	21	-	-	-	-	-	-	-
6	35	30	25	20	15	-	-	-	-	-	-
7	34	29	24	19	14	9	-	-	-	-	-
8	33	28	23	18	13	8	3	-	-	-	-
9	32	27	22	18	14	9	5	0	-	-	-
10	31	26	21	18	13	10	6	3	0	-	-
11	30	25	20	16	13	10	7	4	2	0	-
12	29	24	19	16	12	9	7	4	2	0	0
13	28	23	19	15	12	9	6	4	2	1	0
14	27	22	17	14	11	8	6	4	2	1	0
15	26	21	17	13	10	8	5	3	2	1	0
16	25	20	16	13	10	7	5	3	2	1	0
17	24	19	15	12	9	7	5	3	2	1	0
18	23	18	14	11	8	6	4	3	2	1	0
19	22	17	14	11	8	6	4	2	1	1	1
20	21	16	13	10	7	5	3	2	1	1	0
21	20	15	12	9	7	5	3	2	1	0	0
22	19	15	12	9	6	4	2	1	1	0	0
23	18	15	12	9	6	4	2	1	0	0	0
24	18	15	13	9	7	4	2	1	0	0	0
25	19	15	13	9	7	4	3	1	1	0	0
26	20	15	14	9	8	5	3	2	1	0	0
27	19	14	14	9	9	6	4	2	1	1	0
28	18	13	13	8	8	5	4	3	2	1	0
29	17	12	12	8	8	5	5	2	2	1	0
30	16	11	11	7	6	4	4	2	2	1	0
31	15	10	10	7	7	NP	NP	NP	NP	NP	NP
32	14	10	9	7	5	NP	NP	NP	NP	NP	NP
33	13	11	8	6	6	NP	NP	NP	NP	NP	NP
34	12	11	7	6	6	NP	NP	NP	NP	NP	NP
35	11	11	6	6	6	NP	NP	NP	NP	NP	NP
36	10	10	6	5	5	NP	NP	NP	NP	NP	NP
37	9	9	7	4	4	NP	NP	NP	NP	NP	NP
38	8	8	7	4	3	NP	NP	NP	NP	NP	NP
39	7	7	7	4	3	NP	NP	NP	NP	NP	NP
40	6	6	6	4	2	NP	NP	NP	NP	NP	NP
41	5	5	5	5	3	NP	NP	NP	NP	NP	NP
42	4	4	4	4	3	NP	NP	NP	NP	NP	NP
43	3	3	3	3	3	NP	NP	NP	NP	NP	NP
44	2	2	2	2	2	NP	NP	NP	NP	NP	NP
45	1	1	1	1	1	NP	NP	NP	NP	NP	NP

Table 1.2: Table showing maximal thresholds for Q-gram shapes with regard to size(q) and span of the shape for $m = 50$ and $k = 5$

1.4 CONCLUSION

In order to compute optimal threshold for gapped q-grams we used efficient dynamic programming algorithm and pruning technique as explained in**??** and 1.2 respectively. Even though used DP algorithm is much faster then running all possible miss-match combinations it still has got superpolynomial time complexity with values of m and k.