A 6-frame Translation-based Protein Search Tool

by Yuchen Ge, Zitong He, Mei-Yu Lai, Ruijing Zhang

1. Abstract

Searching translated nucleotide sequences against protein databases has more power identifying the true source organism. In this project, our team implemented a protein search tool based on 6-frame translation of DNA/RNA sequences (*6TBSPs*). We first indexed the input protein references, and then performed local alignment for each translated frame. The output was sorted by e-value, which captures statistical significance. We also designed three test cases, where in the identity test (CDS against protein) *6TBSPs* achieved 100% sensitivity. When BLASTX was benchmarked against our tool, we noticed that 6TBSPs used more query time and memory, but achieved comparable results. Further directions include to improve on the seed-and-extend algorithm to quickly narrow down to candidate regions, as well as making the program parallel using multi-threading.

2. Introduction

With the explosive growth of DNA/RNA sequencing data and expanding capacity of non-redundant protein databases, querying nucleotide sequences against protein becomes useful to infer protein-coding regions and reveal the true identity of source organisms. In this project, we took a **6**-frame **t**ranslation-**b**a**s**ed **p**rotein **s**earch (*6TBSPs*) approach to tackle the molecular sequence determination problem.

Translation is a process by which triplets, i.e., codons, on mRNA are translated into amino acids. However, since the translation initiation site is hidden in DNA, we have to try out all six possible reading frames (3 on each strand) to obtain the desired peptide sequence, thus 6-frame translation. From a biological perspective, protein is more conserved in evolution (i.e., fewer mutations). For homology-based sequence determination, directly searching a translated peptide against protein effectively bypasses potential mutations introduced by DNA, therefore decreasing false positives.

Thanks to advances in high-throughput DNA sequencing technologies, inference of coding sequences (CDS) that was traditionally limited to wet-lab techniques can now be solved computationally using broadly available omics data. Altschul *et al.* (1990) [1] developed the widely cited Basic Local Alignment Search Tool (*BLAST*) and therefore laid the standards for searching molecular sequences. Although *BLAST* is sensitive, its searching speed is comparatively slow [2]. By improving both speed and sensitivity, Steinegger and Söding (2017) [3] developed *MMseqs2*, which is considered as state-of-the-art in protein sequence searching.

In this project, we implemented a search tool, *6TBSPs*, to align DNA sequences against a protein database in all six reading frames. In light of all previous works in the field, and given the duration and scope of this project, we developed a robust tool by performing extensive testing. Specifically, we ensured our tool achieves high sensitivity in discriminating CDS from non-CDS in SARS-CoV-2 and other pathogens, in a reasonable amount of query time. We will also present performance evaluations through a list of thorough test cases as well as benchmark comparisons in later sections.

3. Methods and software

Our software, *6TBSPs*, featured with modular design. In short, we brought a two-stage solution for this problem: 1. protein database construction 2. query with 6-frame translation. The query task was subdivided into several smaller components: 6-frame translation, local alignment, scoring schemes, e-value calculation, backtrack, and output. Each script was written as a function, and then called in the correct order by two main controllers.

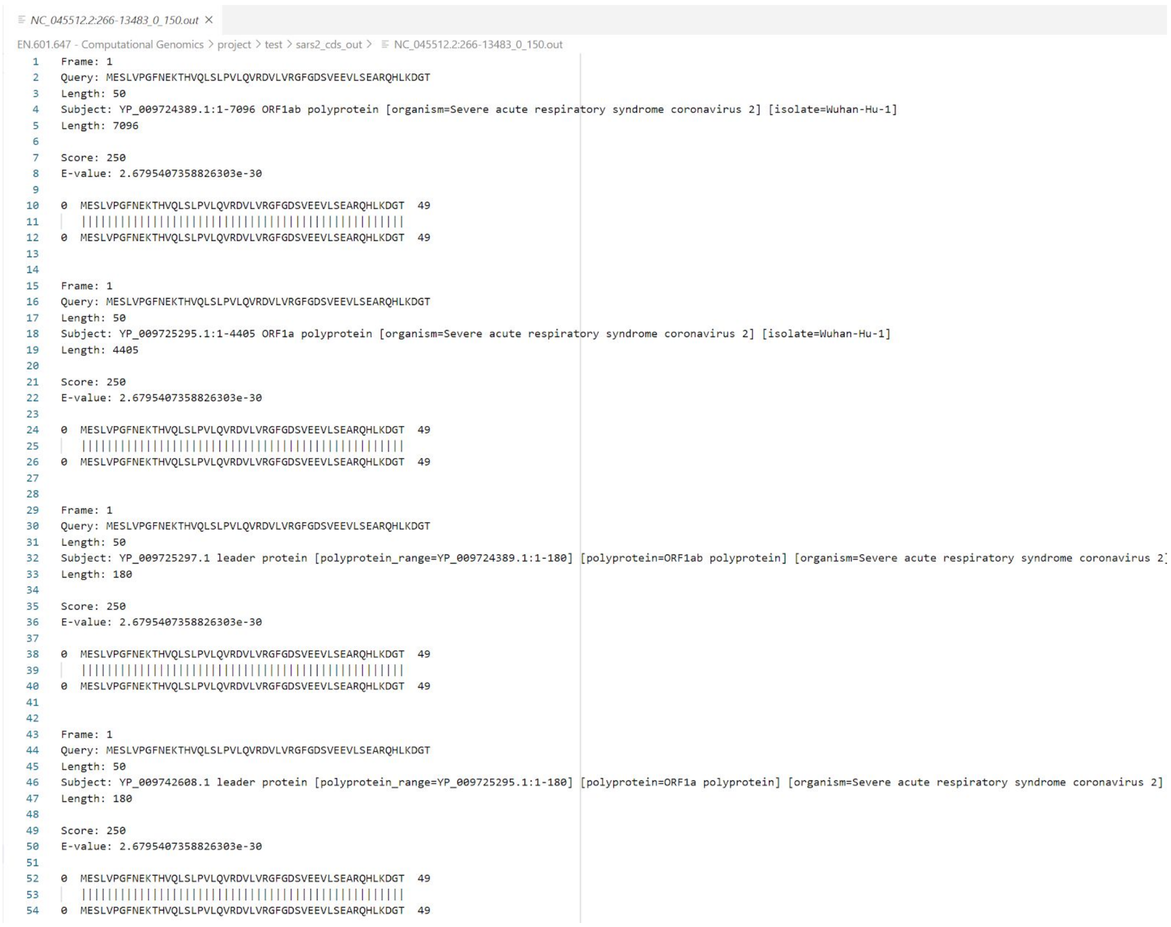
At the top level, *6TBSPs* has two visible programs: *6TBSPs-build* and *6TBSPs-query*. Note that since we decided to take an offline approach, *6TBSPs-build* must be executed before *6TBSPs*-query. *6TBSPs-build* takes in one or more multi-fasta protein sequences as references to be indexed. For smart data structure to facilitate “seed-and-extension”, we utilized the highly-optimized python dictionary class as the backbone data structure to index the protein references. Essentially we stored all possible *k*-mers from the references in a reasonably-sized hashtable, where the key is each unique *k*-mer and the values are tuples of all (accession_id, position) where that *k*-mer (or other *k*-mers with the same hash value) is present. Then the entire dictionary is compressed and flushed to disk using the *pickle* API and later read into memory during query time. As a side note, we stored the reference genomes as well. A simple mechanism was to use 5 bits per amino acid, instead of a normal 1 byte (8 bits) per character.

Once a protein database is properly constructed, *6TBSPs-query* will perform a series of tasks. First the 6-frame translation function takes one or more multi-fasta DNA reads with distinct read ids as input. For each read, the function projects into 6 possible translated peptides, labeled as +1, +2, +3, -1, -2, -3 for future distinction. Each frame of the translated peptide is further broken down into gapped seeds, i.e., non-overlapping *k*-mers with constant gaps. In order to perform fast exact matching, these seeds must have the same length as previously indexed protein reference k-mers. Now we could load the dictionary into the memory and perform a query of exact matches for each seed. If the seeds within the same reading frame result in one or more clusters of hits of certain proximity, we extract from their reference protein only the range covering from the minimum to the maximum position in the same cluster of hits. Then we put the corresponding frame and the extracted regions onto the grid and perform local alignment with affine gap penalty. Per user’s choice, one of the following pre-calculated scoring matrices can be loaded: BLOSUM62 (default), BLOSUM45, BLOSUM50, BLOSUM80, and BLOSUM90 [5], or PAM30, PAM70, and PAM250 [6]. The resulting scores are recorded and plugged into an e-value calculation function [7].

Finally, for each DNA read, the program sorted all protein alignments by e-value (default) regardless of frames and output in ascending order (smallest e-value on the first line) into a summary file, one line per hit. In the output file, we also printed out statistics like the frame number, raw alignment score, e-value, length and range of each pair of query and subject. Last but not least, we called a backtrack on each alignment object, resulting in a 2D alignment graph between each pair of query and subject printed accordingly for visualization.

4. Results

In this section, we highlighted some of the results on the test datasets, to ensure the efficacy and robustness of our software. For all test cases described below, the protein database in-use was constructed from SARS-CoV-2 reference proteins (included in test/SARS2-reference/ncbi_dataset/data/protein.faa).



```
NC_045512.2:266-13483_0_150.out ×
EN.601.647 - Computational Genomics > project > test > sars2_cds_out > NC_045512.2:266-13483_0_150.out
Frame: 1
Query: MESLVPGFNEKTHVQLSLPVLQVRDVLVRGFGDSVEEVLSEARQHLKDGT
Length: 50
Subject: YP_009724389.1:1-7096 ORF1ab polyprotein [organism=Severe acute respiratory syndrome coronavirus 2] [isolate=Wuhan-Hu-1]
Length: 7096

Score: 250
E-value: 2.6795407358826303e-30

0  MESLVPGFNEKTHVQLSLPVLQVRDVLVRGFGDSVEEVLSEARQHLKDGT  49
   ||||||||||||||||||||||||||||||||||||||||||||||||||
0  MESLVPGFNEKTHVQLSLPVLQVRDVLVRGFGDSVEEVLSEARQHLKDGT  49


Frame: 1
Query: MESLVPGFNEKTHVQLSLPVLQVRDVLVRGFGDSVEEVLSEARQHLKDGT
Length: 50
Subject: YP_009725295.1:1-4405 ORF1a polyprotein [organism=Severe acute respiratory syndrome coronavirus 2] [isolate=Wuhan-Hu-1]
Length: 4405

Score: 250
E-value: 2.6795407358826303e-30

0  MESLVPGFNEKTHVQLSLPVLQVRDVLVRGFGDSVEEVLSEARQHLKDGT  49
   ||||||||||||||||||||||||||||||||||||||||||||||||||
0  MESLVPGFNEKTHVQLSLPVLQVRDVLVRGFGDSVEEVLSEARQHLKDGT  49


Frame: 1
Query: MESLVPGFNEKTHVQLSLPVLQVRDVLVRGFGDSVEEVLSEARQHLKDGT
Length: 50
Subject: YP_009725297.1 leader protein [polyprotein_range=YP_009724389.1:1-180] [polyprotein=ORF1ab polyprotein] [organism=Severe acute respiratory syndrome coronavirus 2
Length: 180

Score: 250
E-value: 2.6795407358826303e-30

0  MESLVPGFNEKTHVQLSLPVLQVRDVLVRGFGDSVEEVLSEARQHLKDGT  49
   ||||||||||||||||||||||||||||||||||||||||||||||||||
0  MESLVPGFNEKTHVQLSLPVLQVRDVLVRGFGDSVEEVLSEARQHLKDGT  49


Frame: 1
Query: MESLVPGFNEKTHVQLSLPVLQVRDVLVRGFGDSVEEVLSEARQHLKDGT
Length: 50
Subject: YP_009742608.1 leader protein [polyprotein_range=YP_009725295.1:1-180] [polyprotein=ORF1a polyprotein] [organism=Severe acute respiratory syndrome coronavirus 2]
Length: 180

Score: 250
E-value: 2.6795407358826303e-30

0  MESLVPGFNEKTHVQLSLPVLQVRDVLVRGFGDSVEEVLSEARQHLKDGT  49
   ||||||||||||||||||||||||||||||||||||||||||||||||||
0  MESLVPGFNEKTHVQLSLPVLQVRDVLVRGFGDSVEEVLSEARQHLKDGT  49
```

Figure 1. The read ranges from [0, 150) in NC_045512.2:266-13483 ORF1a polyprotein [organism=Severe acute respiratory syndrome coronavirus 2] [isolate=Wuhan-Hu-1]. When searched against the SARS-CoV-2 protein database, it yields 4 equally best hits, where the second hit corresponds to its protein identity.

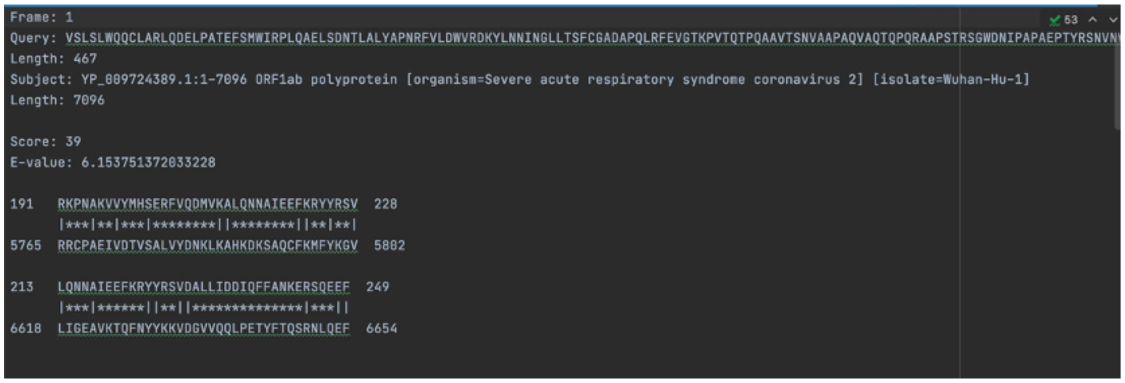
4.1. Test case 1 (identical):

We wrote a customized read simulator to generate error free reads (included in read_simulator.py). For this test, we simulated reads of length 150 and coverage 0.1 from SARS-CoV-2 CDS (included in test/SARS2-reference/ncbi_dataset/data/cds.fna). This resulted in 55 simulated reads from CDS (included in test/sars2_cds_l150_c01.fa). Because these reads were obtained from CDS directly, we expected that every single one of them should have true identity hits against the SARS-CoV-2 reference protein database.

The results confirmed as we got 55/55 true positives. Thus we achieved 100% sensitivity in this case. Figure 1 is an example output of one of the simulated reads named NC_045512.2:266-13483_0_150.

4.2. Test Case 2 (highly dissimilar):

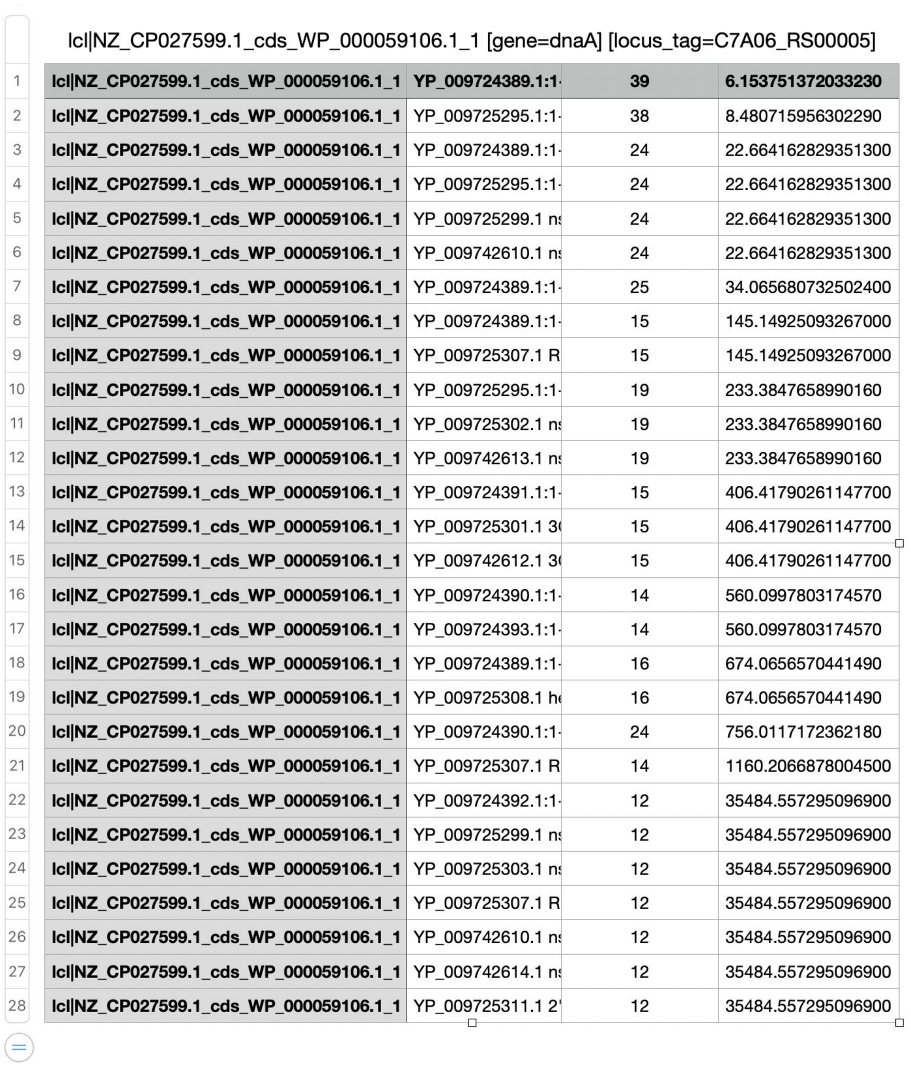
In case 2, we tested the specificity of our 6TBSPs. We randomly chose 10 reads(~1000bp each) from Escherichia coli strain 97-3250 chromosome, and searched them against the SARS-Cov-2 protein database. Given all negative samples, the ideal output should be all with low score and high e-value, which means the alignment is either in short length or matched by chance (for example, repeated CGs).



```
Frame: 1
Query: VSLSLWQQCLARLQDELPATEFSMWIRPLQAELSDNTLALYAPNRFVLDWVRDKYLNNINGLLTSFCGADAPQLRFEVGTKPVTQTPQAAVTSNVAAPAQVAQTQPQRAAPSTRSGWDNIPAPAEPTYRSNVN
Length: 467
Subject: YP_009724389.1:1-7096 ORF1ab polyprotein [organism=Severe acute respiratory syndrome coronavirus 2] [isolate=Wuhan-Hu-1]
Length: 7096

Score: 39
E-value: 6.153751372033228

191   RKPNAKVVYMHSERFVQDMVKALQNNAIEEFKRYYRSV  228
      |***|**|***|********||********||**|**|
5765  RRCPAEIVDTVSALVYDNKLKAHKDKSAQCFKMFYKGV  5802

213   LQNNAIEEFKRYYRSVDALLIDDIQFFANKERSQEEF  249
      |***|******||**||**************|***||
6618  LIGEAVKTQFNYYKKVDGVVQQLPETYFTQSRNLQEF  6654
```

Figure 2. Example output file.

As shown in Figure 2, for the first read, the best hit showed an e-value of 6.15, which obviously lacks biological meaning. For all the ten reads, we didn’t find any hit with an e-value less than 2 and a score exceeding 50 at the same time. This result also got verified from BLAST+ results, which though give a lower e-value for some hits, still far from meeting the criterion for a biologically meaningful hit. In conclusion, the performance of 6TBSPs on specificity was up to expectation.



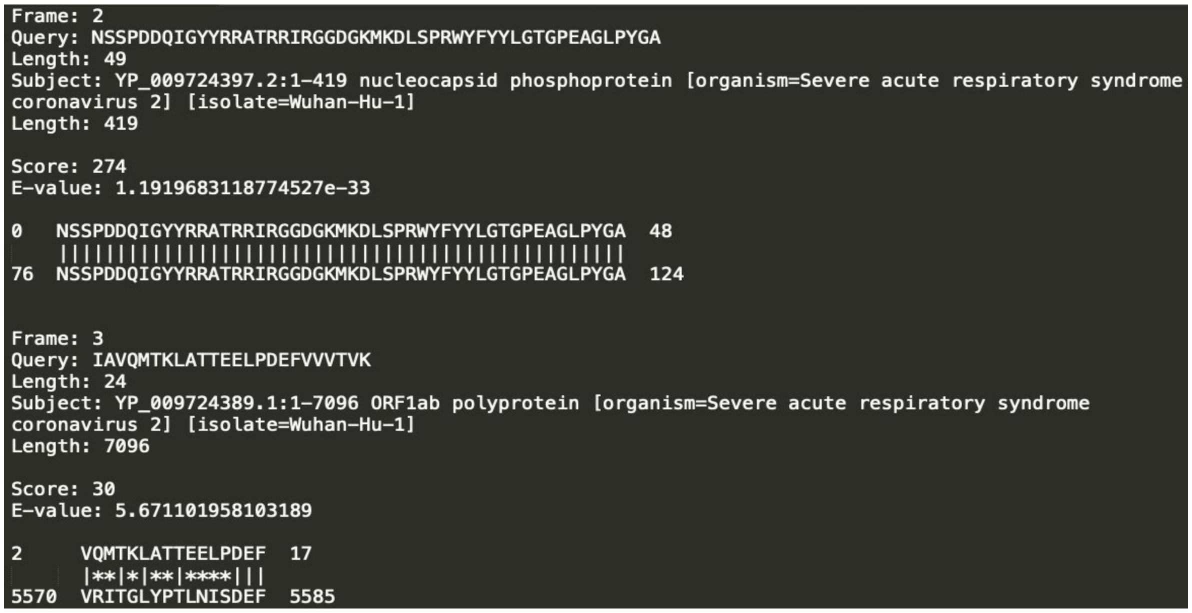
lcl|NZ_CP027599.1_cds_WP_000059106.1_1 [gene=dnaA] [locus_tag=C7A06_RS00005]

1	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	**YP_009724389.1:1-**	**39**	**6.153751372033230**
2	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009725295.1:1-	38	8.480715956302290
3	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009724389.1:1-	24	22.664162829351300
4	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009725295.1:1-	24	22.664162829351300
5	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009725299.1 ns	24	22.664162829351300
6	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009742610.1 ns	24	22.664162829351300
7	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009724389.1:1-	25	34.065680732502400
8	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009724389.1:1-	15	145.14925093267000
9	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009725307.1 R	15	145.14925093267000
10	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009725295.1:1-	19	233.3847658990160
11	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009725302.1 ns	19	233.3847658990160
12	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009742613.1 ns	19	233.3847658990160
13	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009724391.1:1-	15	406.41790261147700
14	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009725301.1 3(	15	406.41790261147700
15	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009742612.1 3(	15	406.41790261147700
16	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009724390.1:1-	14	560.0997803174570
17	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009724393.1:1-	14	560.0997803174570
18	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009724389.1:1-	16	674.0656570441490
19	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009725308.1 h	16	674.0656570441490
20	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009724390.1:1-	24	756.0117172362180
21	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009725307.1 R	14	1160.2066878004500
22	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009724392.1:1-	12	35484.557295096900
23	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009725299.1 ns	12	35484.557295096900
24	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009725303.1 ns	12	35484.557295096900
25	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009725307.1 R	12	35484.557295096900
26	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009742610.1 ns	12	35484.557295096900
27	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009742614.1 ns	12	35484.557295096900
28	**lcl|NZ_CP027599.1_cds_WP_000059106.1_1**	YP_009725311.1 2'	12	35484.557295096900

Figure 3. Example summary file. Columns from left to right: Subject sequence, alignment length e-value

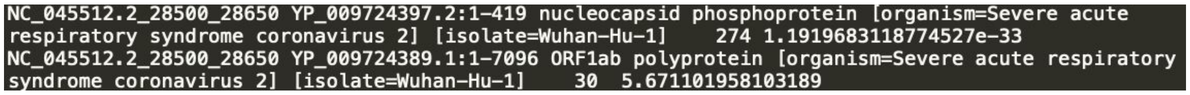
4.3. Test case 3 (metagenomics)

We selected several metagenomics sequences from different human viruses including SARS-CoV-2 to test our software.



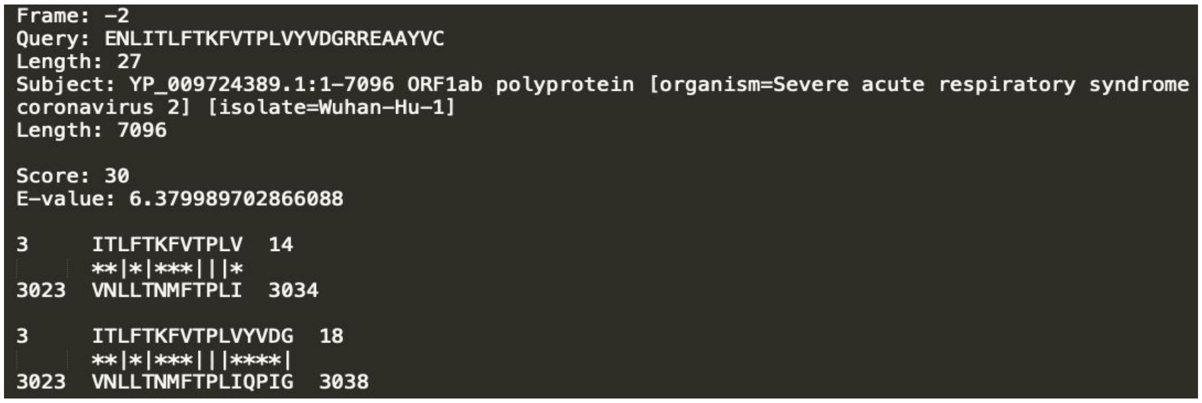
```
Frame: 2
Query: NSSPDDQIGYYRRATRRIRGGDGKMKDLSPRWYFYYLGTGPEAGLPYGA
Length: 49
Subject: YP_009724397.2:1-419 nucleocapsid phosphoprotein [organism=Severe acute respiratory syndrome
coronavirus 2] [isolate=Wuhan-Hu-1]
Length: 419

Score: 274
E-value: 1.1919683118774527e-33

0   NSSPDDQIGYYRRATRRIRGGDGKMKDLSPRWYFYYLGTGPEAGLPYGA  48
    |||||||||||||||||||||||||||||||||||||||||||||||||
76  NSSPDDQIGYYRRATRRIRGGDGKMKDLSPRWYFYYLGTGPEAGLPYGA  124


Frame: 3
Query: IAVQMTKLATTEELPDEFVVVTVK
Length: 24
Subject: YP_009724389.1:1-7096 ORF1ab polyprotein [organism=Severe acute respiratory syndrome
coronavirus 2] [isolate=Wuhan-Hu-1]
Length: 7096

Score: 30
E-value: 5.671101958103189

2     VQMTKLATTEELPDEF  17
      |**|*|**|****|||
5570  VRITGLYPTLNISDEF  5585
```

Figure 4. The outputs of a SARS-CoV-2 sequence in Frame 2 and Frame 3. The sequence in Frame 2 is highly similar with SARS-CoV-2 protein database.



```
NC_045512.2_28500_28650 YP_009724397.2:1-419 nucleocapsid phosphoprotein [organism=Severe acute
respiratory syndrome coronavirus 2] [isolate=Wuhan-Hu-1]    274 1.1919683118774527e-33
NC_045512.2_28500_28650 YP_009724389.1:1-7096 ORF1ab polyprotein [organism=Severe acute respiratory
syndrome coronavirus 2] [isolate=Wuhan-Hu-1]    30  5.671101958103189
```

Figure 5. As expected, the sequence is simulated from SARS-CoV-2 genomes from position [28500, 28650). According to the gene feature annotations, we know that from 28274 to 29533 is a CDS corresponding to gene N, so the e-value must be close to 0 (E-value = 1.1e-33) when we searched against the SARS-CoV-2 protein database.



```
Frame: -2
Query: ENLITLFTKFVTPLVYVDGRREAAYVC
Length: 27
Subject: YP_009724389.1:1-7096 ORF1ab polyprotein [organism=Severe acute respiratory syndrome
coronavirus 2] [isolate=Wuhan-Hu-1]
Length: 7096

Score: 30
E-value: 6.379989702866088

3     ITLFTKFVTPLV  14
      **|*|***|||*
3023  VNLLTNMFTPLI  3034

3     ITLFTKFVTPLVYVDG  18
      **|*|***|||****|
3023  VNLLTNMFTPLIQPIG  3038
```

Figure 6. The output of a sequence of Norovirus in Frame -2. When we aligned the sequence against the SARS-CoV-2 protein database, the percentage of identities is very low (42%, 5/12).

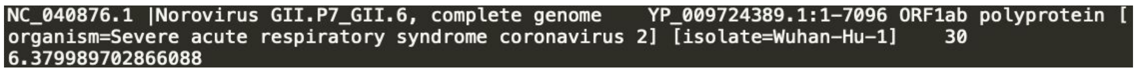

```
NC_040876.1 |Norovirus GII.P7_GII.6, complete genome    YP_009724389.1:1-7096 ORF1ab polyprotein [
organism=Severe acute respiratory syndrome coronavirus 2] [isolate=Wuhan-Hu-1]    30
6.379989702866088
```

Figure 7. The sequence is from Norovirus which is unrelated to SARS-CoV-2, so the e-value is larger than 1 (E-value = 6.38) when we searched against the SARS-CoV-2 protein database.

5. Evaluation - Blastx, ranking loss

We chose proper viral DNA/RNA and protein to simulate the read input and construct the protein database, ensuring the computation scale is sufficient and feasible. Also, we need to ensure *6TBSPs* pass each test case as described in the Objectives section. We benchmarked against *BLASTX* on the same curated dataset in the same computational environment.

5.1. Computational Complexity

We will analyze *6TBSPs* and *BLASTX* theoretically and compare the time complexity and space complexity in big O notation.

We recorded the disk space, peak CPU usage, and query time for each tool. Besides, we ensured the actual performance is proportional to the theoretical complexities and analyze any differences if they exist. Because our implementation is based on Python and wrapped up in a conda environment, which has drawbacks on efficiency compared with C/C++, we will focus on biological correctness rather than actual speed-up.

Table 1: Usage of computing resource and time during building and tests

		6TBSPs	Blastx
Build	Time(s)	User 0.07 + sys 0.04	User 0.02 + sys 0.01
	Peak memory(B)	12476416	3399680
Case1:identical	Time(s)	User 304.57 + sys 2.56	User 0.30 + sys 0.02
	Peak memory(B)	48439296	8220672
Case2: dissimilar	Time(s)	User 95.06 + sys 0.96	User 0.11 + sys 0.02
	Peak memory(B)	48627712	7946240
Case3: metagenomics	Time(s)	User 384.89 + sys 3.12	User 1.35 + sys 0.05
	Peak memory(B)	51396608	26537984

In terms of build time, BLASTX showed high efficiency in database building, but its final database contains 9 subfiles which took some time to be written into the disk. 6TBSPs was a bit slower in database building, but the structure of our database gives only two subfiles.

In terms of query time, BLASTX performed undoubtedly better in query speed. We believed it is related to the number of seeds we selected in our 'seed and extension' step. In our program, for each translated frame we extracted only 2 seeds of length 3 --- one at the start and the other at the very end. Thus it might result in more regions and a much longer range. We need to make a balance between the range of search and the speed of query. However, the speed of 6TBSPs is also quite acceptable for our testing purposes.

6TBSPs' peak memory usage is larger than *BLASTX,* which is developed by C. Considering the Python's object construction will consume more memory space and we used multiprocess in Python to accelerate our tool, this memory usage is reasonable.

5.2. Biological Relevance

We compared the biological relevance of *6TBSPs* with *BLASTX*. Given a query, the tools output sorted results from high score to low score. We expect that if we query a CDS sequence whose corresponding protein is in our curated database, the true hit should be on the top line of the output. However, it's hard for this project to define what exactly a true positive means. Therefore, we introduce a ranking loss term to quantify the performance. We define the ranking loss as below,

$$Ranking\ Loss = \frac{\sum_{i=1}^{n}(R_i - 1)}{n}$$

where R_i is the query's rank in the output table, and n is the total number of reads.

We considered *BLASTX's* output as the 'True Hit' and calculated the *6TBSPs'* ranking loss based on *BLASTX's* output. For a query read, we first extracted the hits which share the lowest e-value from *BLASTX's* output. These hits were considered as top hits. We searched all of the top hits in *6TBSPs'* output and got a bunch of ranks. We chose the minimum rank as R_i in the equation. For each top hit, if there is a tie, such as several of 6TBSP's hits have the same e-value, we choose the lowest rank among these hits.

For those query sequences which don't have top hits in *6TBSP's* output, rank is defined as the number of all hits in the output. The idea is to penalize any false hits that score better than the true hit. In other words, we expect that the true hit scores no worse than any other hits. In this way, we can compare our tool with BLASTX.

We calculated the ranking loss for case 1 and case 3, and count the Miss/Hit for both cases in Table 2.

Table 2: Evaluation Record with respect to BLASTX's output

	Disagree	Agree	Ranking Loss
Test Case 1	0	55	0
Test Case 3	6	36	3.961

For case 1, the ranking loss is 0 and all of the queries get the correct hits, which is our expectation. For case 3, the ranking loss and the count of difference is small enough to reflect the ratio of the number of metagenomics sequences from different human viruses. Therefore, our tools are comparable with *BLASTX*.

6. Conclusion

In this project, we implemented 6TBSPs to realize a 6-frame-translation approach to search DNA/RNA against proteins. We passed and rationalized all three test cases. In test case1 for sensitivity, we found all 55 reads with true protein identities. In test case 2 for specificity, among the 10 negative/unrelated reads, none of them gave a significant e-value. In test case 3 for mixed viral reads, we noticed our tool sensitively discriminated SARS-Cov2 CDS from non-CDS reads.

Overall, we are pleased with the test results. However, there are several points to be improved in the future. First, the seed-and-extend algorithm has capacity for improvement. In the project, we naively chose 2 seeds spanning the entire query. However, if more seeds were selected, more false positives would be filtered out. Second, ideally we would like to parallelize multiple input queries using multi-threading, which possibly reduced query time more, as compared to the gold standard BLASTX.

Contributions:
All of us discussed together to conceptualize this project. Also we contributed equally in terms of drafting the proposal and final report. And, each of us contributed to:

- YG: 6tbsps-build.py, file_io.py, 6tbsps-query.py, seed_and_extend.py; test case 1
- ML: 6tbsps-build.py, file_io.py, six_frame_translation.py, and test case 1,2,3
- RZ: score_matrix.py, e_value_calculation() and test Case 2
- ZH: local_alignment_affine.py, local_alignment_linear.py, evaluator.py, 6tbsps-query.py, score_matrix.py, evaluation

References:

1. Altschul, S. F., Gish, W., Miller, W., Myers, E. W., & Lipman, D. J. (1990). Basic local alignment search tool. *Journal of molecular biology*, *215*(3), 403-410.
2. Huson, D. H., & Xie, C. (2014). A poor man's BLASTX—high-throughput metagenomic protein database search using PAUDA. *Bioinformatics*, *30*(1), 38-39.
3. Steinegger, M., & Söding, J. (2017). MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature biotechnology*, *35*(11), 1026-1028.
4. Pagh, R., & Rodler, F. F. (2004). Cuckoo hashing. *Journal of Algorithms*, *51*(2), 122-144.
5. Henikoff, S., & Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, *89*(22), 10915-10919.
6. Dayhoff, M. O. (Ed.). (1972). *Atlas of protein sequence and structure*. National Biomedical Research Foundation.
7. Karlin, S., & Altschul, S. F. (1990). Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proceedings of the National Academy of Sciences*, *87*(6), 2264-2268.