Shahjalal University of Science & Technology
Dept. Of Computer Science and Engineering

An Approach To Measure Quality among Multiple Alignments and Align Oxford Nanopore Single Molecule Sequence Reads

By
Faisal Ahmed
Reg. No.: 2011331047

Supervisor

Mr. Biswapriyo Chakrabarty
Lecturer, CSE, SUST

An Approach To Measure Quality among Multiple Alignments and Align Oxford Nanopore Single Molecule Sequence Reads

By

Faisal Ahmed

Reg. No.: 2011331047

Supervisor

Mr. Biswapriyo Chakrabarty
Lecturer, CSE, SUST

Co-Advisors

Md. Ruhul Amin Shajib
Assistant Professor, CSE, SUST

Recommendation Letter from Thesis Supervisor

This Student Faisal Ahmed whose thesis entitled **An Approach To Measure Quality among Multiple Alignments and Align Oxford Nanopore Single Molecule Sequence Reads** is under my supervision and agree to submit for examination.

Mr. Biswapriyo Chakrabarty
Lecturer,
Dept. of Computer Science & Engineering,
Shahjalal University of Science & Technology

Date : 19 / 10 / 2016

Qualification Form of Bachelor Degree

Student Name : Faisal Ahmed

Thesis Title : An Approach To Measure Quality among Multiple Alignments and Align Oxford Nanopore Single Molecule Sequence Reads

This is to certify that the thesis submitted by the student named above in 19 October, 2016. It is qualified and approved by the following persons and committee.

....	....	....
Head of the Dept.	**Chairman, Exam Committee**	**Supervisor**
Dr Mohammad Reza Selim	Dr Mohammad Reza Selim	Mr. Biswapriyo Chakrabarty
Professor	Professor	Lecturer

Abstract

Quality measurement of an alignment is an important task in the field of sequence alignment. Due to the advancement in genome sequencing now it is possible to sequence about 18,000 human genome per year. Thus for finding common areas in a genome or aligning coverage is playing a vital role to find common disease genes patterns. Sequence alignment mapping / binary alignment mapping file is a standard format to show result of a specific tool that aligns several reads with a reference genome. There are several tools like IGV, Qualimap to evaluate those files. Qualimap version 2 is the latest version of Qualimap which has outperformed all other tools with various features and statistics giving on input SAM / BAM file. We tried to understand what a SAM file contains and implemented a naïve algorithm named Edit Distance to align multiple short reads with a corresponding reference genome and compared result of those reads with our alignment procedure. For several reads we have been able to outperform our alignment with the dataset of SAM file we are experimenting. But a lot more optimization can be made which will be focused on our future work approach.

Our dataset is from Oxford Nanopore Single Molecule Sequence Reads. Existing tool to align these reads with the reference genome E. Coli is named NanoBlaster. Algorithm of NanoBlaster is experimented with variations and result has been developed on the alignment.

Acknowledgments

I would like to thank my supervisor Mr. Biswapriyo Chakrabarty and my co-advisor Md. Ruhul Amin Shajib for their advice and co-operation toward me during my thesis.

List of Figures:

Contents

Chapter 1

Introduction

1.1 Next Generation Sequencing Alignment

Due to the advancement of Next Generation Sequencing methods cost to sequencing a genome has dramatically fallen where on the other hand output data has risen to a whole new level. Scientists are now highly interested to evaluate these sequences and finding various patterns among them. Here comes the alignment concepts. From then various alignment methods have been introduced and they give various types of alignment with finding important patterns between reads and corresponding reference genome. The target is to try to align a read with a reference genome in a way that matches maximum area in a possible way. Thus we can find structural or evolutionary similarity between them. Sequence Alignment Mapping / Binary Alignment Mapping are file format concept to arrange multiple read alignment information in a file. So we can evaluate that file to measure quality of that particular alignment method considering different criteria and factors.

As different methods can create biases among alignment data of same multiple reads and reference genome, it has become obvious to measure quality of them. Therefore a standard file format has been introduced to keep the data of different alignment method based on their biased factors. Now we can evaluate the alignments and measure quality among them with simply using Sequence Alignment Mapping / Binary Alignment Mapping file data.

1.2 Quality Control of Alignments

Quality control is a much more needed criteria of any alignment. Alignment can be done in a numerous way such as following a naïve edit distance or going through noisy areas with skipping or jumping. Such methods create different achievements or failure in fulfilling alignment criteria. Some methods are memory efficient, some can be time efficient. Basically statistical data need to shown to evaluate and measure minimum quality. Several tools are introduced to evaluate Sequence Alignment Mapping / Binary Alignment Mapping file and giving valuable information on them. They lack in fulfilling different criteria or making an alignment better in different ways.

1.3 Tools To Measure Quality

There are several tools have been developed by scientist to evaluate any alignment through Sequence Alignment Mapping or Binary Alignment Mapping file named Picard tools, RNA SeQC, RSeQC, Qualimap. They evaluate SAM or BAM files differently on different factors. The last tool mentioned here Qualimap is the latest edition in all these tools and uses various information gained on SAM or BAM files to get the all of an alignment.

As per we know for now that these tools don't use extra algorithm to make another alignment to make it more standard, we will try to make an exception here by implementing a naïve algorithm to get more standard alignment. A formal algorithm called Edit Distance can be implemented here for the sake of get a standard scale. To make it more linear we have tried to implement a scoring module here. Our target is to get the maximum matching between a read and a reference genome.

In edit distance algorithm there are four cases which can be got. They are- Insertion, Deletion, Matches and Mismatches. A matrix will be made based on scoring and all possible solution module. We have implemented Dynamic Programming (Bottom Up) here to make the matrix more believable and trustable. We will get a path through bottom-left to top-right. That will tell us exact alignment that has been possible so far in the best case. To keep the path linear scoring distribution plays a vital role. So that to get the maximum match the matching score has been given a positive number and others are given negative numbers so that our penalty will be reduced which is the primary goal.

1.4 Approach To Naive Dynamic Programming

Dynamic Programming is an approach to get the best solution covering all possible best or worst cases. Edit distance can be implemented with dynamic programming in two variations. One is bottom-up and the other one is top-down. We have implemented both these ways and compared. They have given almost same result. Because its' always a linear solution what we will get. As bottom-up don't make stacks like the top-down module it is more time and memory efficient theoretically. So we prefer bottom-up in the most cases.

After implementing edit-distance we traverse through our solution matrix where on one side reference genome is taken and on the other side a sample read is taken. We traversed from top-right to bottom-left to get the actual alignment based on algorithm module. We will describe the algorithm later on the report.

To show differences among the SAM described alignment and the standard scale that we created which is not so memory and time efficient we built some statistical histogram with python. As the implementation of the algorithm was not done completely by that time the histogram shows sample SAM file gives better alignments on maximum reads. For now we have detected some problems about the implementation of the naïve solution as to cut the exact length can be a backward criteria for us to get more standard solution.

How much matches we got through optimal alignments or naïve alignment approaches we must define a scale. We have measured these standards with percentage of identity (POI). It measures percentage of matches considering all the penalties (insertion, deletion, mismatches) it has made. As this is a naïve approach to get the all possible solution and best case by that far we can't consider very lengthy reads that will consume memory and time. Percentage of identity is calculated dividing total matches by total sum of matches, mismatches, insertion and deletions.

Traversing the path might be done with variations. Starting the path from top-right to bottom-left is called global alignment. But there is another kind of alignment here that can also be done. This is called local alignment. We can start the path here from any cell out from the row or column. This will bias the total alignment by cutting a particular part from the read or the reference genome which is not standard but we can observe different things like as what if we try to align a particular part of the read or the read with a particular part of the reference genome. This can give very interesting results like whether a read is more likely to align with a particular part or which part. In pattern finding local alignment plays a vital role more than of the global alignment.

1.5 Evaluating SAM File

Understanding the sequence alignment mapping or binary alignment mapping file is key playing role here in our approach. A better understanding of the file format will make us more comfortable to evaluate it. In short it is called SAM. Different necessary flags indicate different things such as whether the read is revere compliment or not, whether is mapped or not, the chromosome name, read reference name, how this is mapped, number of mismatches, number of matches, number of deletions, number of insertions, position from reference length (this indicates from which position of the reference genome the alignment has been done). Position can be 1-based or 0-based (this is also vaguely included in the file). It also contains the information like whether all the reads are aligned here with the same reference genome or not, next position of the reference or the read. These are some compulsory fields that must be maintained for the file format. There are some other optional fields too like matching scores (where we can know the actual information about the used algorithm and matching criteria on the alignment).

Chapter 2

Background and Related Work

Sequence alignment is playing a significant role in BioInformatics. It gives us the information about how a DNA is similar to other DNAs'. Human genome is a vast area of secrets. Scientists have been unveiling them since DNA had been explored. A DNA is a complete code. It has all the information of all the genes and everything. When we try to match a dna with others we need a to know in which particular regions they are matching and their matching percentage. A genome contains all the information about all the genes and alleles.

When a genome is aligned with another genome there can be three kind of operations- matches, mismatches and indels. Indels mean insertion and deletion. As sequence can be very large so we got to keep the alignment procedure in an optimized way. There are many algorithms for alignments. Some are memory efficient, some are time efficient. Standard algorithm is likely to be dynamically programmed. The goal is to achieve maximum matching. Dynamic programming algorithms are not memory efficient. So various techniques have been worked on various datasets to work on a memory and time efficient method. Alignment representation is another important task. To know where the matches are, where the mismatches are, where the inserts are, where the deletions are we need to represent the alignment data with a proper way.

A way can be a graph to show the noise or a 2D array whether matching points are increased or decreased, penalties are increased or decreased linearly. Scoring in alignment is very important. Score can be distributed in many ways according to the variation of the particular algorithm. There are various types of alignment algorithms.

Two main kinds are local alignment and global alignment. Local alignment means when a read is partially aligned with another read. It can be started from any position of a read rather than from its start to end with whole read. Nowadays with numerous data alignment plays a vital role with different types and variations. Global alignment means aligning the total read with from both starts to both ends. The score is counted at the corner end of 2D score matrix here. Name of the method of different types of alignments are, pairwise alignment, dot-matrix alignment, dynamic programming, word method. If we consider to alignment on more than two sequences then we call it multiple sequence alignment. There are different methods to do multiple sequence alignment. They are, dynamic programming, progressive method, iterative methods, motif finding. Dynamic programming is the most reliable for sequence alignment as its error rate is most low. It ensures most error-free, most matching alignment between 2 reads. There are some disadvantages on dynamic programming. It is not memory and time efficient. For very long reads dynamic programming a failed way to do alignment. On the other hand block alignment procedure is both time and memory efficient but error rate is high here. As it is not optimal it goes with the higher score of a block and jump on to another block.

Dynamic programming works with the pairwise alignment. Otherwise say for multiple sequences it doesn't work. Insertion means if in the alignment processing characters need to be inserted to get the optimized high score, and the deletion works similarly but in the opposite direction like deleting characters to get the optimized high matching score. To represent alignment dot-matrix alignment is another process. It is easy to visualize the performance of an alignment algorithm in a dot matrix process but not to get the result. What dot matrix does is pointing a dot on the matching character row and column wise. So we can get easily the visualisation where density of matching is high and where mismatches are high. Thus we can add lines to that will result for either insertions or deletions. In very large scale databases word method is used. It will not confirm optimally best result.

Scientists have developed various types of software with various types of applications within them. Blast stands for basic local alignment search tool. It makes its search with fast k-tuple. And this process is heuristic. CS-Blast stands for context specific basic local alignment search tool. It is more sensitive than the previously developed blast. The dynamic programming which is vastly used for optimized less-error alignment is called smith waterman algorithm. Cudasw++ is a tool which uses this algorithm on its base and it is accelerated on the gpu with multiple shared gpu hosts. Double indexing algorithm is used on diamond tool which works on blastx and blastp. Fasta tool also works as blast does, but it is less sensitive than blast and works as local search with fast k tuple with heuristic search process. For pairwise alignment there are other tools also. Acana is one them. Alignme is for protein alignment. BioPerl, Bioconductor both follow dynamic programming procedure. There are tools that use multiple GPUs' to work with very long sequences. Cudalign is one of them. Blastsz, lastz work as pattern matching among seeds. Mutiple protein sequence alignment comparison is done statistical significance information is done by compass. Segment based method is followed by dialign-tx and dialign-t. We know there are various information hidden on sequences. To model these information mAlign tool helps. It helps to model the information contained on an alignment. Needlemen-wunsch dynamic programming method works on needle tool. NW tool works also on needlemen-wunsch dynamic programming algorithm. Gpas uses backtracking technique with gpu and dynamic programming combination. Ssearch uses smith-waterman dynamic programming algorithm but it also gives statistics about the alignment.

Needleman-wunnsch dynamic programming algorithm is followed by stretcher tool with much more memory optimization advancement. There is a tool which aligns nucleic acid sequences with having a protein alignment, the name of the tool is tranalign. Msa, msaprobes, multalin, multilagan they all use dynamic programming either in a progressive way or in a non-progressive way. There are many other ways to make alignment among multiple sequencing like using hash tables and making cache memory optimization. With probabilistic consistency pecan plays a great role on multiple sequence alignment. Besides that, genetic algorithm using goes in trend with saga which stands for sequence alignment with genetic algorithm. RevTrans combines protein and dna alignment with back translating the protein alignment to dna. Tcoffee produces more sensitive progressive alignment.

With long length strings, short reads have become popular also with its significances on genomic research works. In alignment tools managing and considering gap opening in other words insertions and deletions penalty plays a great role. Some tools maintain it and create varieties in results. BarraCUDA is one of them. It accelerates GPGPU and implies burrows-wheeler transform short read alignment program. BBMap is another short read alignment tools which is more sensitive and specific than burros-wheeler aligner. Affine transform optimized global alignment is its implemented algorithm. It's a bit slower process but gives better result than smith-waterman well defined dynamic programming algorithm. By indexing k-mers from reference sequence and reads we can make alignment faster and more accurate. But it is not memory efficient and it can be overcome with hash tables by hashing the k-mer reads.

BFAST tool applies indexed reference sequence reads to align short reads. It can also handle deletions, insertions and colour errors. It uses smith-waterman dynamic programming algorithm on its core. BigBWA tool uses a hadoop cluster to use burrows-wheeler-aligner. There may have variations on BWA algorithm with BWA-ALN, BWA-SW, BWA-MEM. It reduces computational time complexity and add scalability and fault tolerance. Blastn uses sequence database and not accurate. Its time complexity is also high. American programmer and research scientist Jim Kent also developed a tool named BLAT to align short reads. It can handle one mismatch initially. Bowtie tool can align approximately 25 million reads in 1 cpu hour time. It creates 1.3 gb footprint for human genome and permanent indexes which are re-usable. It uses burrows-wheeler transform algorithm. Hive-hexagon tool at first uses a hash table and a bloom matrix and thus it creates and filter potential position on the whole genome reference sequence. It also uses cross similarity between short reads and avoids realigning non unique redundant sequences. It procedure is faster than normal bowtie and bwa. It also allows insertions and deletions and divergent sensitive alignment on viruses and more conservative eukaryotic alignments. Position specific scoring practice also known as its short form pssm is used by bwa-pssm tool. It's a probabilistic short read alignment tool. It can consider quality scores of the reads and models data biases. Cloudburst also uses Hadoop to reduce maps to map short reads. Cuda-ec is an error correcting short read aligner with using error correction gpu. Cashx can handle a large amount of short read sequence data. It contains a set of tool that can be used together as running as independent modules. Its' used algorithm's accuracy rate is very high for perfect hits to a reference genome.

Chapter 3

Current Technology

The first thing we have to do to evaluate a specific alignment algorithm is to evaluate a sam film. SAM stands for sequence alignment mapping. It maps alignments for a great vast or variety of reads with a genome sequence. There are tools to evaluate a sam file. But they are not so memory efficient. Normally a sam file is of large size life 2-3 GB. That's why processing such file needs memory efficient programs. Qualimap is a tool which is used to read data from sam file and gives output on them. Like what percentage has been mapped with the main genome sequence and how the alignments have been gone with the reads.

Qualimap software is written with R and Java language and it runs on multithreads. It takes input bam file. A bam file is a binary representation of sam file. It stands for binary alignment mapping. It gives several charts and statistics as output so that those statistical analysis on charts can be used in future for further researches. There are other tools also named rna-seqc, rseqc, integrative genomic viewer (igv). Firstly the program splits the reference genome sequence into several parts of windows of several sizes say the size is 400 here. The for every window parts it goes down to the depth of analysis like reading counts and other things.

Users can also see a dedicated panel that has summarized results and can use a general feature format or a browser extensible data to make attention of specific regions of interest areas. Qualimap outputs a number of charts, some of them are, coverage percentage vs length. That is how much percentage of the genome sequence has been covered by the reads with the alignment algorithm and its length wise frequencies. Insertion deletions mismatches counting with the scale of base-pairs is another operation. It can count length wise length wise frequencies of insertions deletions and mismatches and shows them in a chart for the users. As the sequence depth grows higher the number of features detected gets also high with this. Drawing saturation plot is one the important feature plots to get features with more sequence depth. Picard tool is also one of tools to read sam/bam files. It deals with various ngs applications. Its interface is command line where qualimaps user interface is a gui. Rseqc's interface is also a command line. On the other hand rna-seqc's user interface is also gui. With command lines we can also work on qualimap and rna-seqc. Picard tool outputs some raw data plots. Different picard tool creates different types of plots. Qualimap outputs a summary report. The report is created with html or pdf. Rna-seqc and rseqc give alignment statistics partially. But qualimap and picard tools give full alignment statistics.

SAM stands for sequence alignment mapping. It is a file format which contains all the information of a specific alignment algorithm when it aligns a reference genome with a bunch of reads. How the reads were aligned and number of mismatches, insertions, deletions all the important statistics can be derived from a sam file. BAM stands for binary alignment mapping. It binary format of a corresponding sam file. Qualimap and other tools like qualimap just read data from sam or bam file as input and gives output with numerous types of statistics, graphs and charts. A sam file maintains a standard like how it should be written as it is a standard format to map alignment data.

We will discuss now different type of flags a sam file contains in it. QNAME is the first flag that is contained by a sam file. It's a string. It defines the name of the query template. RNAME is also a string flag which gives the reference genome name. Then comes POS which tells us the location of the reference genome sequence where the alignment starts to align the particular read. RNEXT is the name of next reference genome for the alignment of next read sequence. PNEXT is the position of the next read. TLEN is the length of the observed template from the reference genome. There are other Int data type flags are there. Different Int numbers indicate different things. 1

means a template which has multiple segments in alignment sequencing. 2 means every segment has been aligned properly. This information is according to the alignment algorithm which we also refer as our aligner. 4 means a segment that has not been mapped yet, that means it has remained as unmapped. 8 means the sequence of the next segment is unmapped. 16 means this segment must be reveres complemented to get the final output. 32 means the sequence of the next segment has to be reversely complemented to get the final output. 64 means this is the first read of the corresponding sam formatted template. 128 means that is the last sequence reading of that particular sam formatted tamplate. QUAL flag plays a very important role in the sam file. It tells the quality threshold of the aligner corresponding to a particular reading with the reference genome.

We have analysed a recent tool which was prepared to align Oxford Nanopore Single Molecule Sequence Reads. The main interesting point with this dataset is that this dataset is highly noisy and densely. What this tool does is applying a bunch of methods simultaneously in a different approach. So that we can make a good alignment with e. coli and these reads. To read the alignment result which was written in a sam formatted file we faced problems as we are shortage of ram. So we first built a tool to examine the alignment result got from the sam formatted file. 1000 number of reads were taken from the sam file and their lengths are from 1 to 1000. Then we made alignment with the e. coli and these reads. Each read was started from the position mentioned in the sam file. Sam file did mention particular positions for each and every reads to align them with the reference genome sequence. Our tool aligned with the naïve dynamic programming algorithm. The complexity of the algorithm is high. It's not memory and time efficient. We built this tool for the only purpose to get the quality of the alignment of the sam file so that we can understand of the quality of the alignment that exists.

Nanoblaster is an existing tool to do the job of aligning oxford nanopore highly noisy dataset with the e. coli which is our main reference genome sequence read. It first makes some pair wise k-kmers. Then this k-mers are aligned with making a specific size of window distance. Then there will be some unaligned data base pairs. These unaligned base pairs are clustered with their neighbour k-mers. This is done with block alignment processing. Blocks are advanced with maximum scoring from the corner matrix.

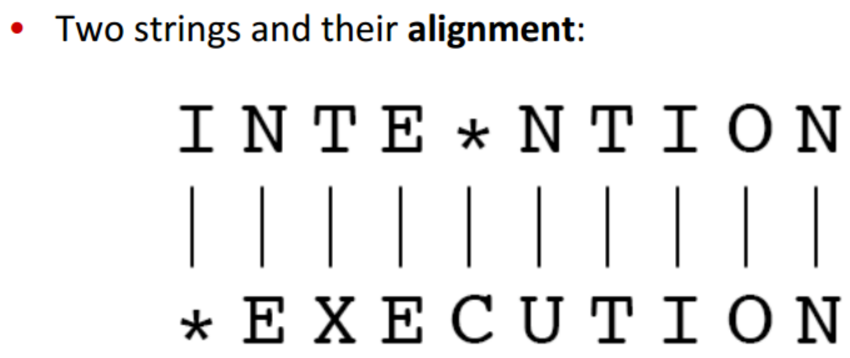
Chapter 4

Methodology

Here we will discuss what we have implemented so far to construct a standard scale.

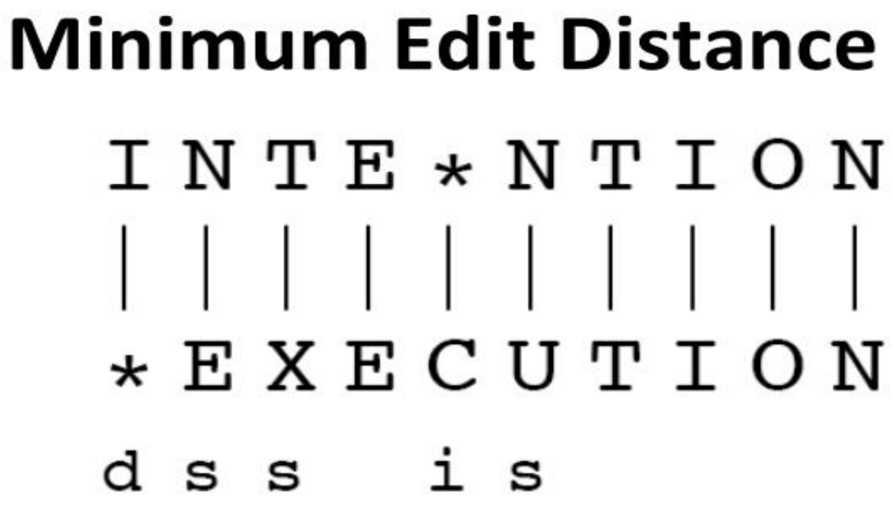
Firstly we have implement a well-known string matching algorithm called “Edit Distance”. This tells us how it will match or mismatch between a read and a reference genome. This algorithm can be implemented in two ways. As we have to find the optimal result we have to implement it in such a way that ensures all possible solutions and get the best out of it. Basically this algorithm can be implemented two ways. One is bottom up approach and the other one is top down approach. Top down approach is a recursive method. On the other hand bottom up doesn’t need to be recursive so it is more memory efficient as it doesn’t take stacks to find the solutions. We have implemented it both way but the result was almost same as the scoring system ensured it would be a linear solution.

For example we can see below two figure that has aligned a read with a reference string. The upper one is reference and the other one is read.



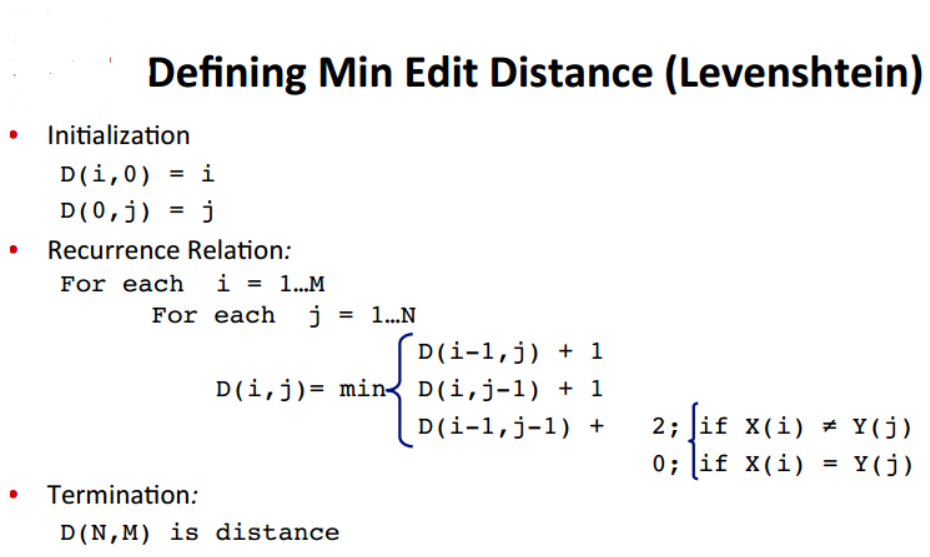
- Two strings and their **alignment**:

```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
```



Here d means deletion, i means insertion, s means mismatches and the other ones are matching.

Alignment is basically of two types with edit distance with respect to path finding. The below figure shows the sudocode of the algorithm.



Defining Min Edit Distance (Levenshtein)

- Initialization
```
D(i,0) = i
D(0,j) = j
```
- Recurrence Relation:
```
For each  i = 1…M
      For each  j = 1…N
                            D(i-1,j) + 1
          D(i,j)= min{      D(i,j-1) + 1
                            D(i-1,j-1) +   2; if X(i) ≠ Y(j)
                                           0; if X(i) = Y(j)
```
- Termination:
```
D(N,M) is distance
```

The Edit Distance Table

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1									
#	**0**	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

The Edit Distance Table

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1									
#	**0**	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases}$$

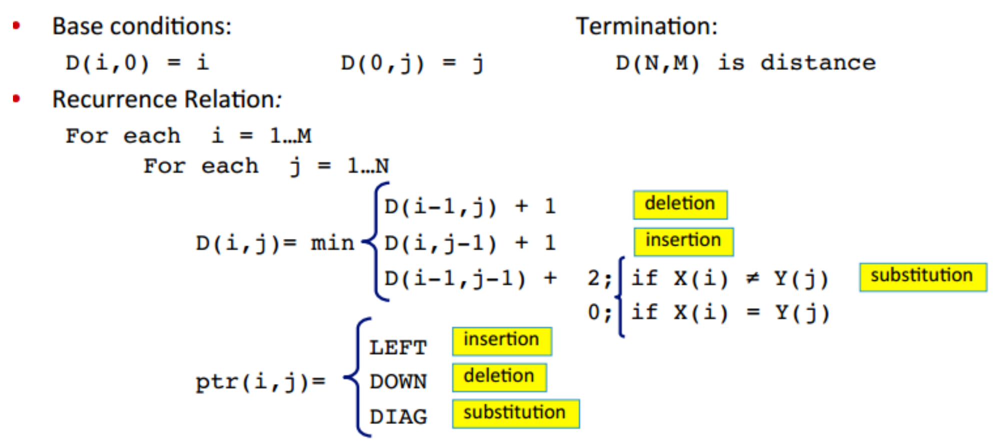
The Edit Distance Table

N	9	8	9	10	11	12	11	10	9	**8**
O	8	7	8	9	10	11	10	9	8	9
I	7	6	7	8	9	10	9	8	9	10
T	6	5	6	7	8	9	8	9	10	11
N	5	4	5	6	7	8	9	10	11	10
E	4	3	4	5	6	7	8	9	10	9
T	3	4	5	6	7	8	7	8	9	8
N	2	3	4	5	6	7	8	7	8	7
I	1	2	3	4	5	6	7	6	7	8
#	**0**	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

Here the blue ones are indication that we should follow this pathway to find the optimally best reference string that has matched.

n	9	↓ 8	↙←↓ 9	↙←↓ 10	↙←↓ 11	↙←↓ 12	↓ 11	↓ 10	↓ 9	↙ **8**
o	8	↓ 7	↙←↓ 8	↙←↓ 9	↙←↓ 10	↙←↓ 11	↓ 10	↓ 9	↙ **8**	← 9
i	7	↓ 6	↙←↓ 7	↙←↓ 8	↙←↓ 9	↙←↓ 10	↓ 9	↙ **8**	← 9	← 10
t	6	↓ 5	↙←↓ 6	↙←↓ 7	↙←↓ 8	↙←↓ 9	↙ **8**	← 9	← 10	←↓ 11
n	5	↓ 4	↙←↓ 5	↙←↓ 6	↙←↓ 7	↙←↓ **8**	↙←↓ 9	↙←↓ 10	↙←↓ 11	↙↓ 10
e	4	↙ 3	← 4	↙← **5**	← **6**	← 7	←↓ 8	↙←↓ 9	↙←↓ 10	↓ 9
t	3	↙←↓ 4	↙←↓ **5**	↙←↓ 6	↙←↓ 7	↙←↓ 8	↙ 7	←↓ 8	↙←↓ 9	↓ 8
n	2	↙←↓ **3**	↙←↓ 4	↙←↓ 5	↙←↓ 6	↙←↓ 7	↙←↓ 8	↓ 7	↙←↓ 8	↙ 7
i	**1**	↙←↓ 2	↙←↓ 3	↙←↓ 4	↙←↓ 5	↙←↓ 6	↙←↓ 7	↙ 6	← 7	← 8
#	**0**	1	2	3	4	5	6	7	8	9
	#	**e**	**x**	**e**	**c**	**u**	**t**	**i**	**o**	**n**

Following is the path finding back-tracing algorithm.



- Base conditions: Termination:

```
D(i,0) = i        D(0,j) = j        D(N,M) is distance
```

- Recurrence Relation:

```
For each  i = 1…M
     For each  j = 1…N
```

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 & \text{deletion} \\ D(i,j-1) + 1 & \text{insertion} \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } X(i) \neq Y(j) \\ 0; & \text{if } X(i) = Y(j) \end{cases} & \text{substitution} \end{cases}$$

$$ptr(i,j) = \begin{cases} \text{LEFT} & \text{insertion} \\ \text{DOWN} & \text{deletion} \\ \text{DIAG} & \text{substitution} \end{cases}$$

This type of path printing is called global alignment, but if we choose any particular index from row or column then it is called local alignment. We have found some good results from this alignment from our implementation that will be shown in result and discussion chapter.

Chapter 5

Experiment

There is a tool named NanoBlaster that has been developed to align data from Oxford Nanopore Single Molecule Sequencing Reads. We have tried to make some experiment on it. The result was good so far. At first, we have taken the reference genome in consideration. Then we mapped it in different k-mers. Here we have experimented with k=(11,12,13,…). Then again for per read we mapped them with same process.

Now we have mapped data (starting position of every k-mer in the reference genome and all the reads). A variable named threshold is used in our experiment which will conclude a k-mer from reads data if maximum threshold numbers of mismatches been found or not.

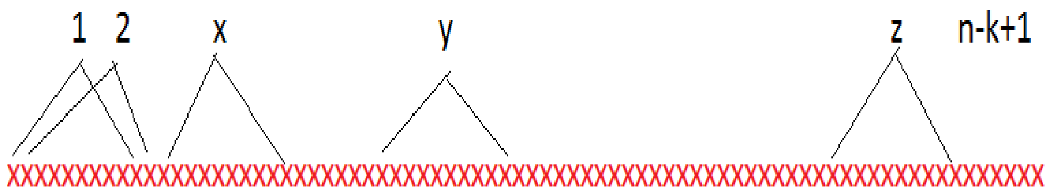
For every read data we call our solve function. Which first merge reference genome mapped data and reads mapped data first in a 2-D vector. After doing that have a 2-D vector from which we can find which k-mer from reference genome can be aligned with which k-mer in the read data.

Next step is to call 2-D LIS function. LIS stands from longest increasing subsequence. This is a traditional algorithm from which we can find the increasing pairs of sequence.

After getting the LIS result we merge the reference genome with the read from the 2-D LIS vector data we got in the earlier step.

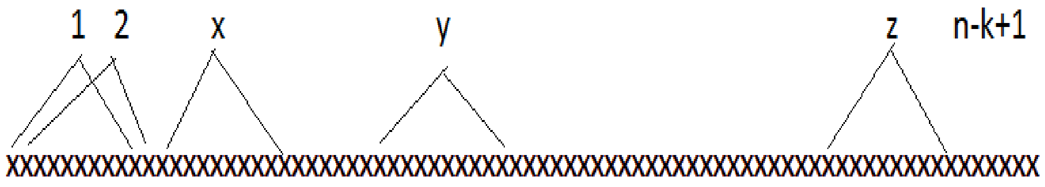
After merging them we have other data which have not been merged from reference genome and read. So we need to break them down here with marking them as different substring pairs which should be aligned.

To align the unmerged substring pairs we go to a different way rather than putting them in 2-D LIS method. We can have a pair of strings and make k*k matrices and make alignment on them. The visualisation of the total process is given below with figures.



1	xxxxxxx
2	xxxxxxx
3	xxxxxxx
4	xxxxxxx
5	xxxxxxx
n-k-1	xxxxxxx
n-k	xxxxxxx
n-k+1	xxxxxxx

Fig: Reference Genome Mapping Indexing


1
2
x
y
z
n-k+1
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

1	xxxxxxx
2	xxxxxxx
3	xxxxxxx
4	xxxxxxx
5	xxxxxxx
n-k-1	xxxxxxx
n-k	xxxxxxx
n-k+1	xxxxxxx

Fig: A Read Mapping Indexing

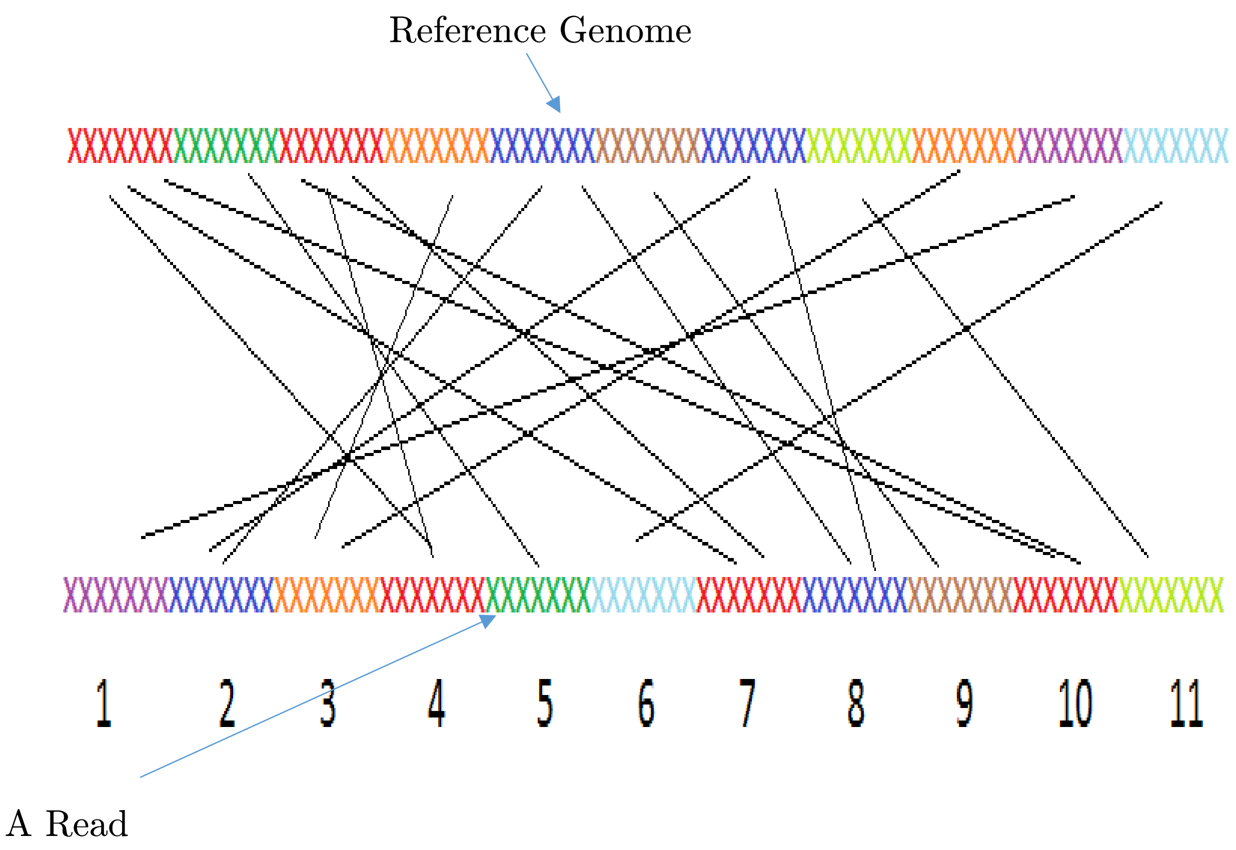


Fig: Merging A List of K-mers from Reference Genome with A list of K-mers from a Read

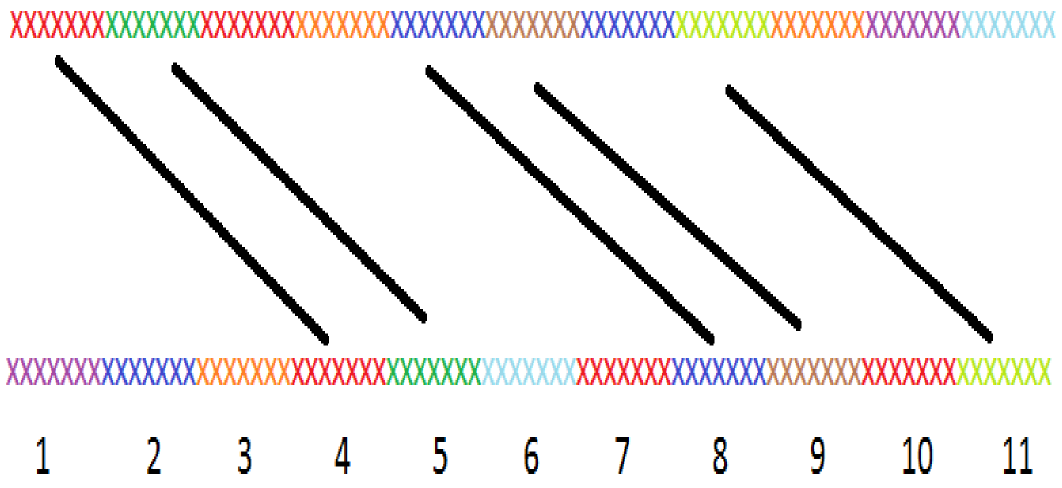
K-Mer starting positon from Reference Genome	K-Mer starting position from a Read
1	4
1	7

1	10
2	5
3	4
3	7
3	10
4	3
5	2
5	8
6	9
7	2
7	8
8	11
9	3
10	1
11	6

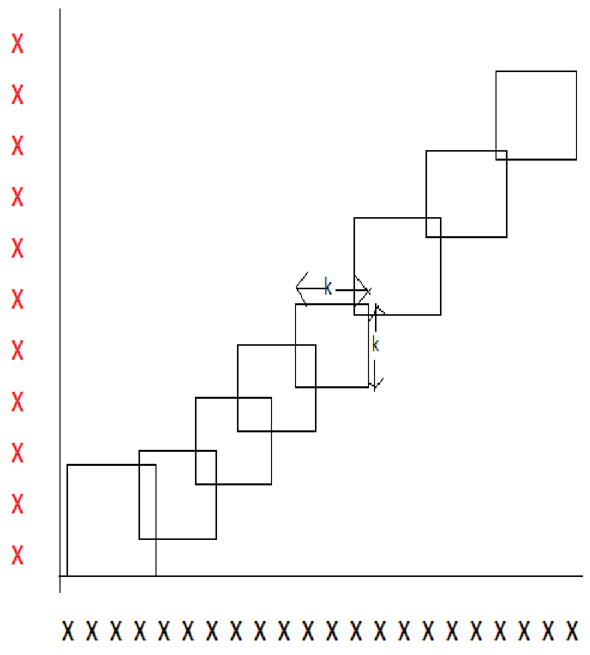
Table: Merging them in a table for the 2-D LIS Input

After 2-D LIS we get,

K-Mer starting positon from Reference Genome	K-Mer starting position from a Read
1	4
2	5
5	8
6	9
8	11



As we can see here, there are strings that has not been aligned. So we will try to do block alignment with k*k size blocks over there.



Chapter 6

Result and Discussion

For data we have got a SAM file which includes thousands of reads. We took 1000 reads from there with maximum length of 1000. Results from NanoBlaster tool.

After alignment the result has four things to consider to give it a score. They are number of matches, number of mismatches, number of insertions and number of deletions.

Percentage of identity is a well-known established scale to measure quality of a particular alignment performance.

Percentage of identity = Total number of matches / Summation of total matches, mismatches, insertions and deletions.

Following figures show what we have experimented and what result we have found so far.

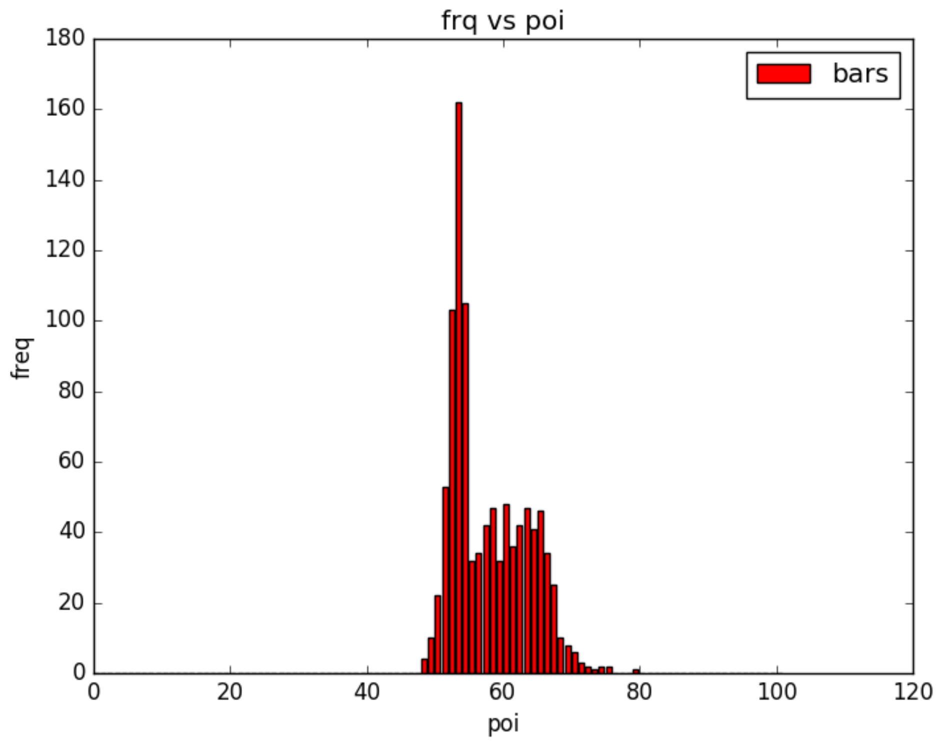


Fig 1: On this figure on x-axis percentage of identity is drawn and on y-axis their frequency is drawn.

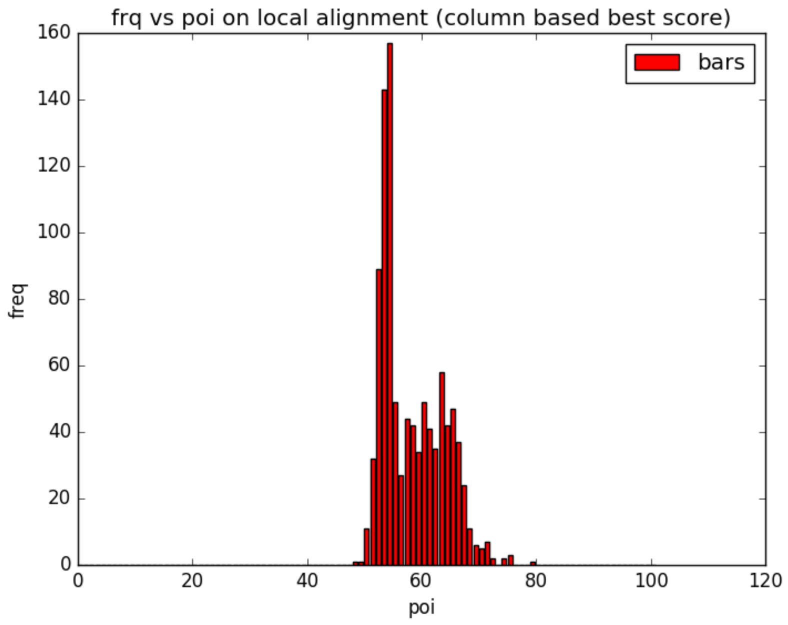


Fig 2: This figure shows local alignment result on column best score finding.

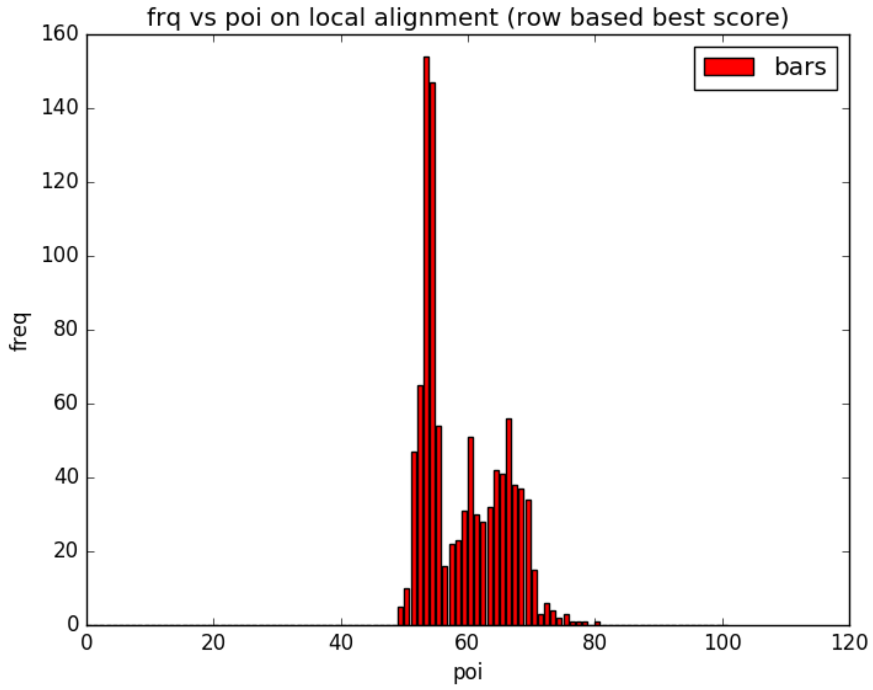


Fig 3: This figure shows local alignment result on row best score finding.

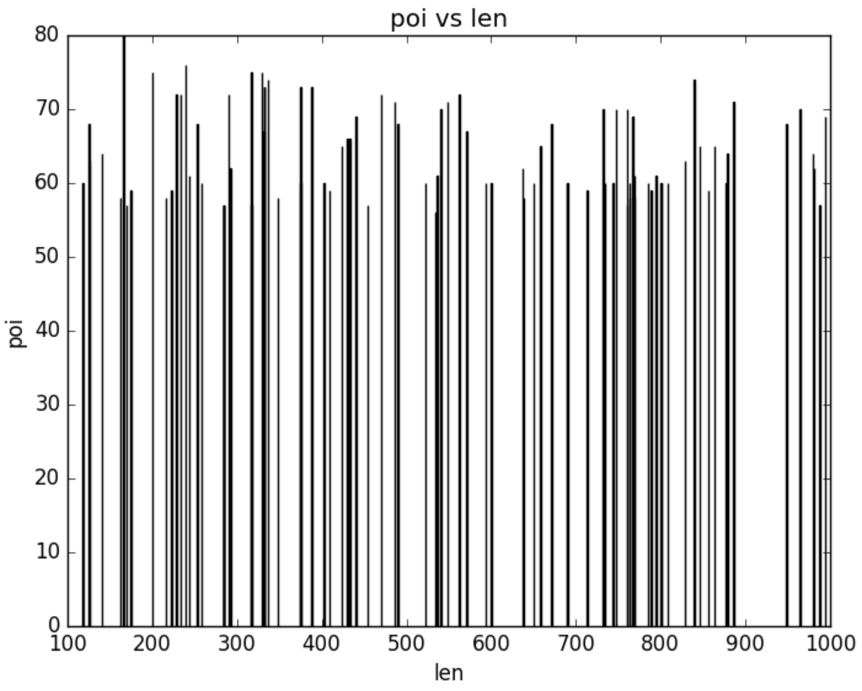


Fig 4: This figure shows what best poi has we got so far with respect to length of reads.

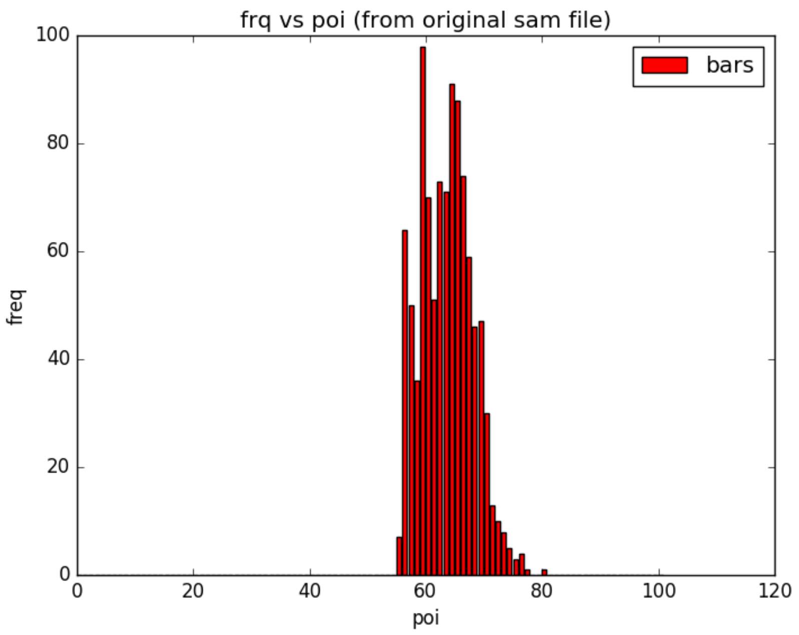


Fig 5: This result is got from original SAM file data.

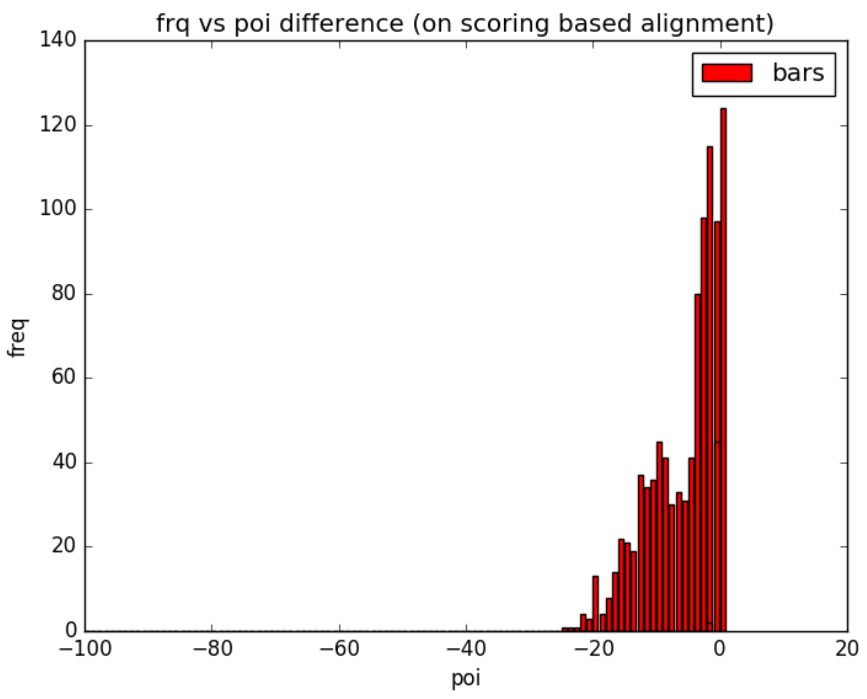


Fig 6: POI difference with global alignment

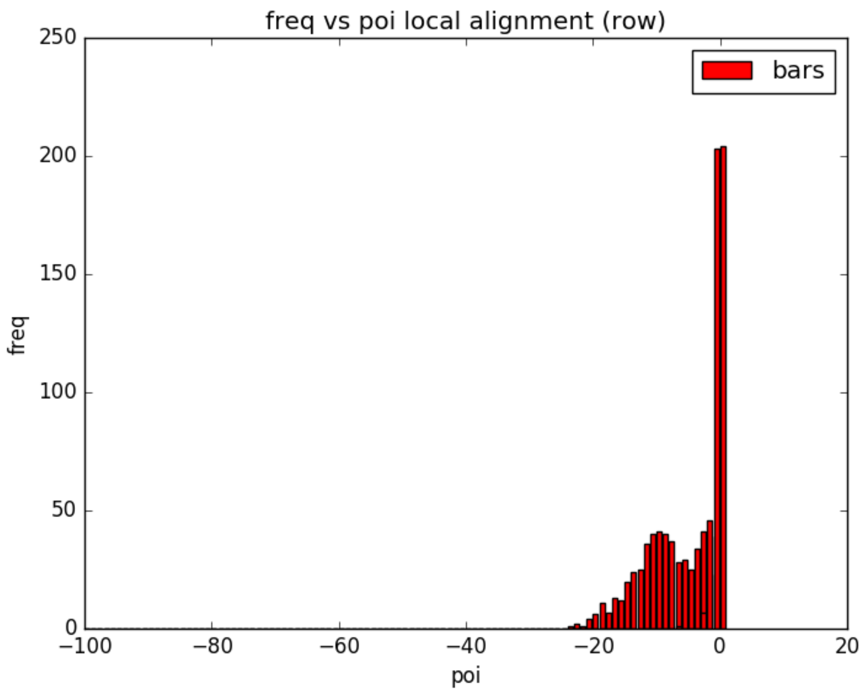


Fig 7: POI difference with local alignment (row)

Now we will analyze some graph statistics of the result got from our experimented method. Some POI score we got here is 80 and best result we got from k-mer window size when it is 11 and 13. For k-mer sizes 12, 14 and others the threshold value didn't make very good score.

Results with 2D LIS and Block Alignment:

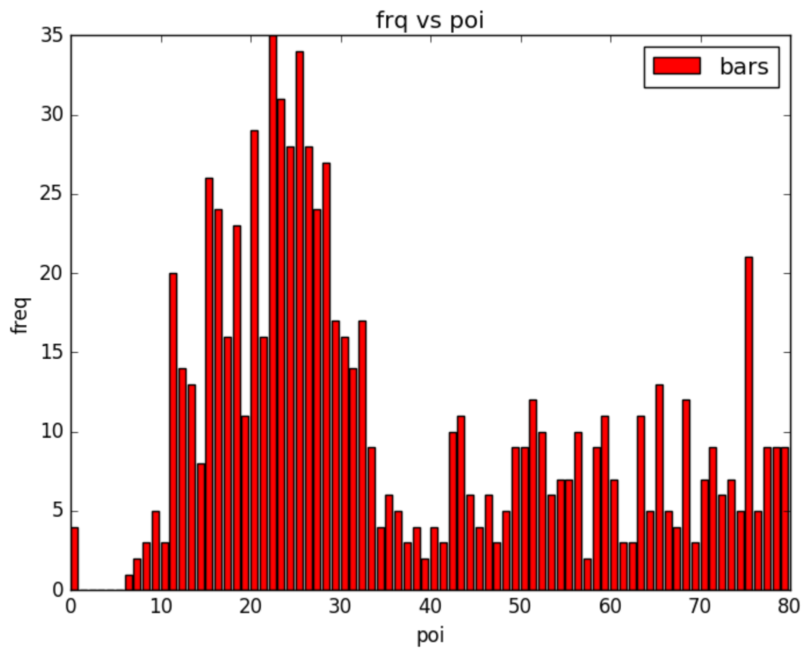


Fig 8: For k=11

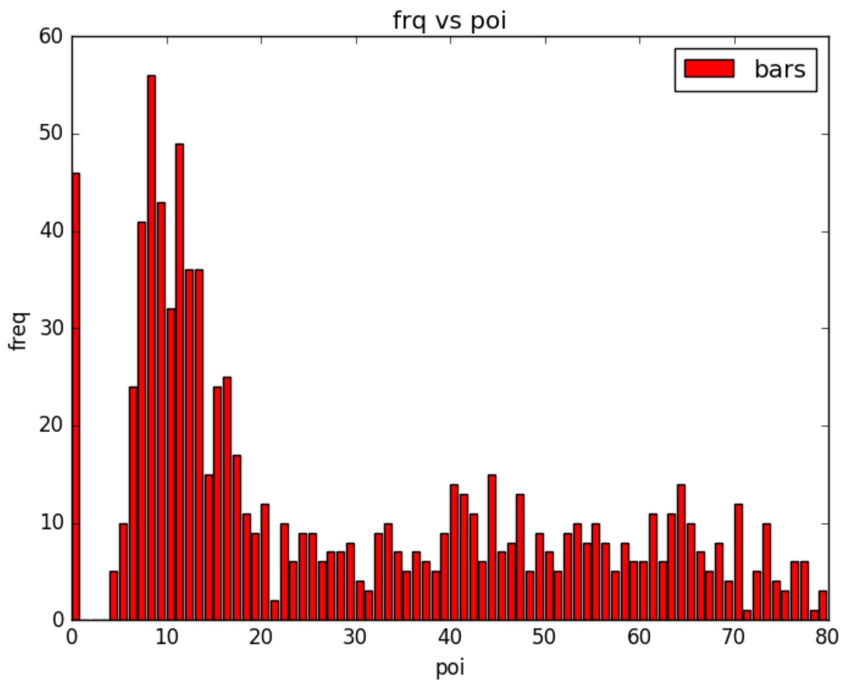


Fig 9: For k=12

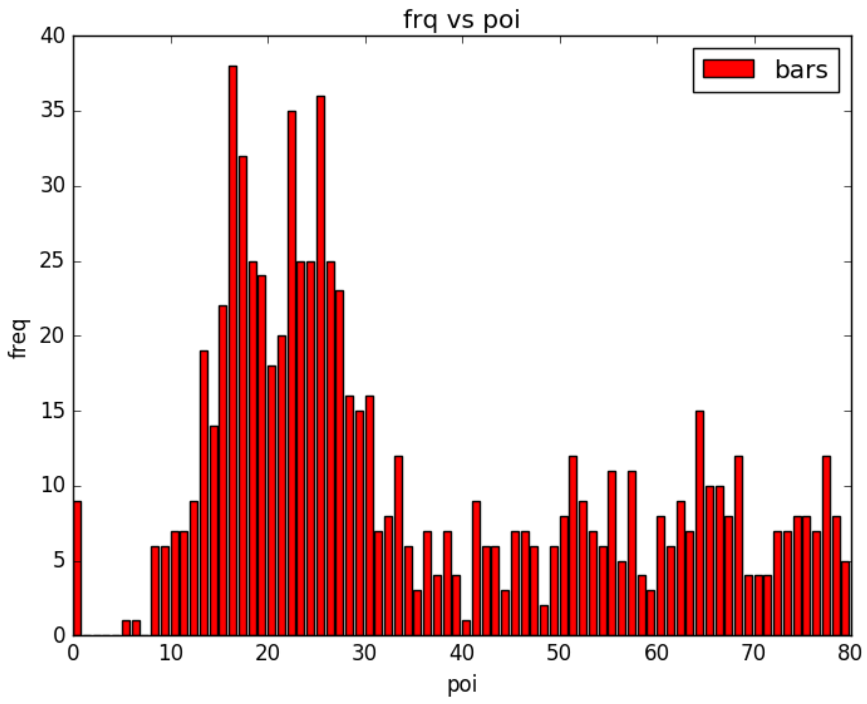


Fig 10: For k=13

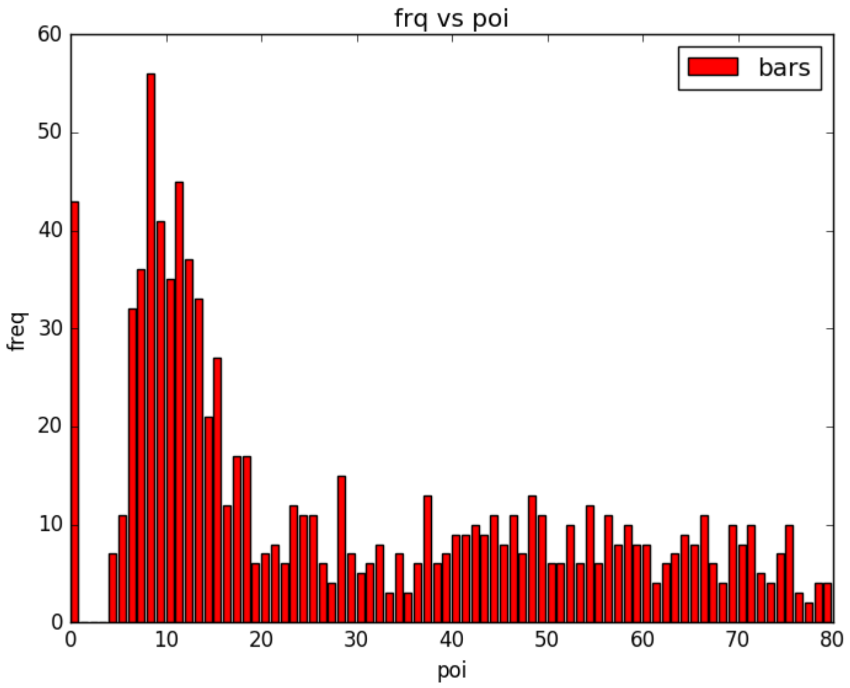


Fig 11: For k=14

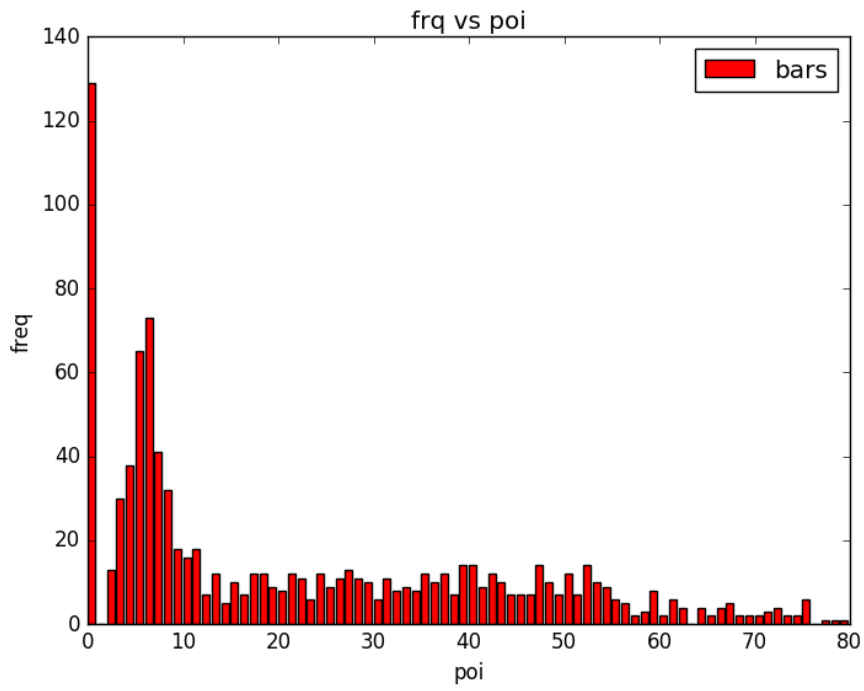


Fig 12: For k=15

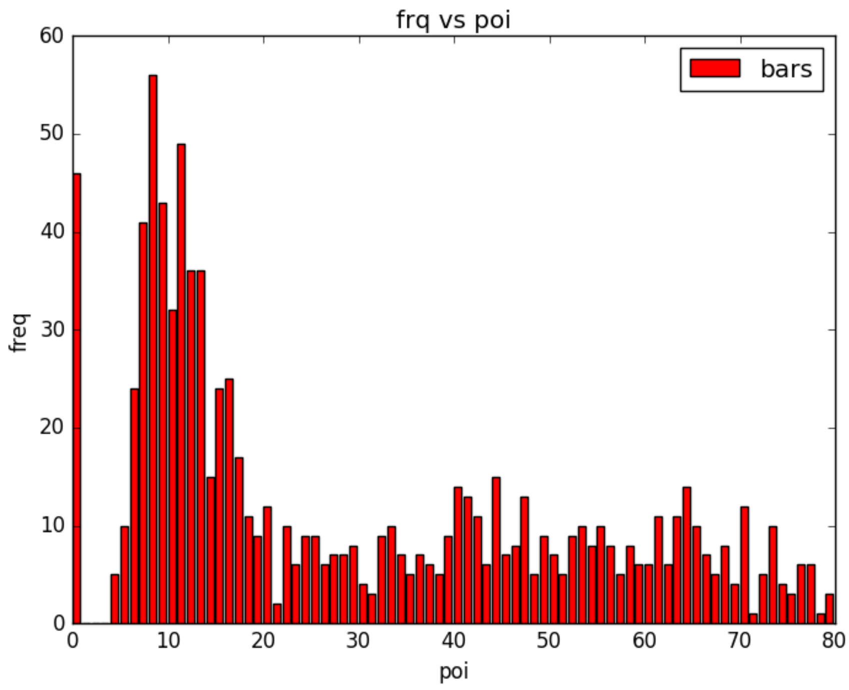


Fig 13: k=16

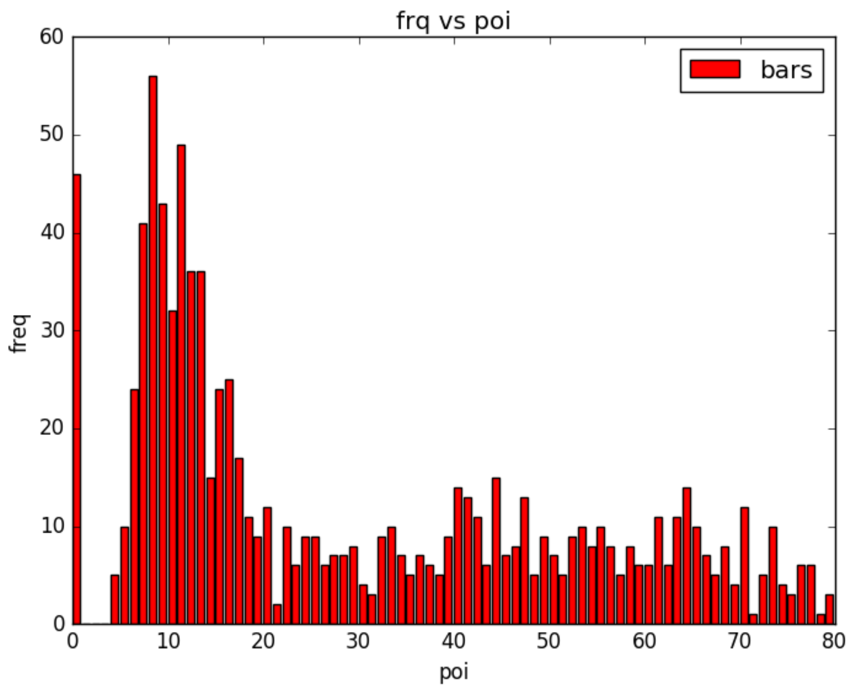


Fig 14: k=17

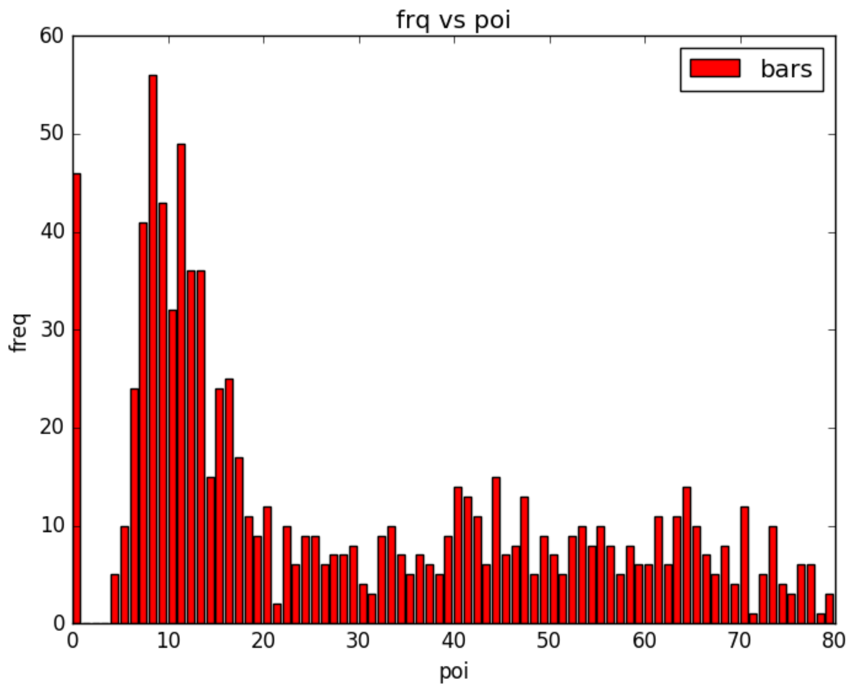


Fig 15: k=18

Chapter 7

Future Work Proposal

Our naive approach has not worked well for many reads. We will try for the betterment of that with more optimized window sized alignment. So that will be able to evaluate SAM file results with more error free alignments.

SAM file evaluating tools have many backwardness as Qualimap tool doesn't apply any algorithm to evaluate the reads. It just gives several statistics on a particular SAM file. We will go for the comparison of more than one SAM file on same reads and their alignments on different algorithms. Then we will be able to have a clear view how those reads differ with different positions (deletions, insertions, matches, mismatches) and how it can be improved with having more analysis on particular reads with reference genome.

Experiment methods k-mer finding complexity is not time and memory efficient. So pre-processing of k-mer pairs from reference genome and sequence reads should be faster. So that the complexity will be much more efficient than this.

Conclusion

Quality measurement of alignment is as important as to validate and compare a specific alignment algorithm with others. So that better approaches will get their recognition in a standard scale.

More and better approaches can be experimented to align these dataset. Alignment plays a very important role in bio-informatics. Pattern finding and other similar fields rely on it. Memory and time efficient different alignment methods give different results on different datasets.

References

1. A Compression Model for DNA Multiple Sequence Alignment Blocks. Luís M. O. Matos, Diogo Pratas, and Armando J. Pinho

2. Pairwise sequence alignment algorithms: a survey, Waqar Haque, Alex Aravind, Bharath Reddy

3. Qualimap : Evaluating next-generation sequencing alignment data. García-Alcalde F, Okonechnikov K, Carbonell J, Cruz LM, Gotz S, Tarazona S, Dopazo J, Meyer TF, Conesa A.

4. Qualimap 2: advanced multi-sample quality control for high-throughput sequencing data, Konstantin Okonechnikov, Ana Conesa, Fernando Garcia-Alcalde

5. Dna sequencing with chain-terminating inhibitors. Sanger, F., Nicklen, S., Coulson, A.R.

6. Domain enhanced lookup time accelerated BLAST. Grzegorz M Boratyn , Alejandro A Schaffer, Richa Agarwala, Stephen F Altschul, David J Lipman and Thomas L Madden

7. Improved data analysis for the minion nanopore sequencer. Jain, M., Fiddes, I.T., Miga, K.H., Olsen, H.E., Paten, B., Akeson, M

8. Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. Berlin, K., Koren, S., Chin, C.-S., Drake, J.P., Landolin, J.M., Phillippy,