

1 Experiments

To access efficiency of both algorithms, and their MapReduce frameworks set of experiments was performed on the data just described. Two different forms of tests were performed; some for measuring the precision of the algorithms, and some for measuring the speed of the algorithms.

As it is only desired to test the speed of **MM** and $\text{MM}_{\frac{1}{2}}$ on RNA and DNA, only input of the same form as the **fasta** file format will be accepted as input. For all experiments, $\epsilon = 0.95$. The **MM** clustering algorithm will be referred to as **MM**, likewise will the $\text{MM}_{\frac{1}{2}}$ clustering algorithm be referred to as $\text{MM}_{\frac{1}{2}}$. All results from UCLUST was retrieved using the `usearch8.0.1517_win32` version, using the command `cluster_fast input.fa -id 0.95 -uc clusters.file`. As both **MM** and $\text{MM}_{\frac{1}{2}}$ are required to store the clusters, it would be more accurate to demand the same of UCLUST, which is what UCLUST does onto `clusters.file` using the above command.

1.1 Precision tests

In Eq. ??, a metric of the error was set up for the number of hash functions H . This error did not provide an answer to relation between H and the size of k -mers k and its error. Therefore, a practical testing of this relation was performed, to investigate the precision of **MM** and $\text{MM}_{\frac{1}{2}}$.

A gold standard was instrumental to determine the precision of our algorithms. For this purpose, the Levenshtein similarity was used, as defined in Section ??. If two strings had a Levenshtein similarity above ϵ they were placed in the same cluster. The gold standard, **GS**, were then set to be the number of clusters found by the Levenshtein similarity clustering algorithm. Once the **GS** were retrieved, the error of an algorithms' found number of clusters, C , could be defined as the difference between C and the gold standard **GS**

$$E = |C - \mathbf{GS}|$$

similarly, the percentage of error E_p was

$$E_p = \frac{|C - \mathbf{GS}|}{\mathbf{GS}} \quad (1)$$

The lower E was, the more precise the algorithm. As both **MM** and $\text{MM}_{\frac{1}{2}}$ results depend on k and H , a large set of tests were performed to determine the precision of each algorithm at many k -mer sizes k and number of hash functions H . When the optimal settings were found, the results could then be compared to the precision of the UCLUST algorithm.

1.1.1 MM precision tests

First, precision tests of the **MM** algorithm were made. In order to narrow down the number of k and H to test, a heat map of a wide set of k and H for seeing the big scope was made, as seen in Fig. 1. From this wide study, ostensibly the error increased rapidly at $k > 13$, and $k < 5$ were less precise than $k = 5$. Also, at $H > 80$ the error seemed to stop changing significantly.

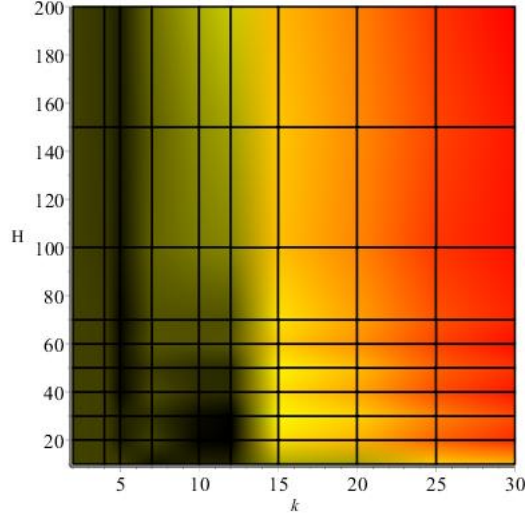


Figure 1: Heat map of the error of **MM** at a widely dispersed array of k -mer sizes k and number of hash functions H , run on **Cecum1**. Black signifies the smallest error, red the highest error, and yellow medium error relative to black and red.

The results of this test showed that the more rigorous tests could be performed at a more narrow set of k and H , as seen in Fig. 2. All heat maps showed very similar results for all samples. There were specific areas that seemed to repeatedly have a very low error, specifically at

- $k=5$, a black line from $H = 50$ to $H = 80$.
- $k = 6$, two black spots at around $H = 10$ and $H = 22$.
- $k = 10, 11, 12$, two big black spots at around $H = 16$ and $H = 30$.

These areas of interest were deemed to be most precise. Therefore, each was tested alongside **UCLUST**, to be able to compare them to each other. Fig. 3 shows the results of these tests.

Fig. 3(a) shows that at $k = 5$, the overall performance of the algorithm appeared near comparable to **UCLUST**. At the spikes, the precision was even higher than **UCLUST**, most notably at $H = 54$, $H = 65$ and $H = 66$, where the average error over all samples can be read from the graph to be approximately 4%.

Fig. 3(b) shows that at $k = 6$, the maximal error is around 70%, much worse than that of **UCLUST**. However, two spikes also appeared here at $H = 13$ and $H = 22$, both with an average error of around 7%, slightly better than **UCLUST**.

Fig. 3(c-e) show that in spite of generally portraying a high error overall at $k = 10, 11, 12$, reaching maximum errors of over 80%, they still had potentially low errors; Most notably at $k = 10, H = 20$, $k = 11, H = 30$, and $k = 12, H = 30$, with spikes with an average error about 10%, about the same

as UCLUST.

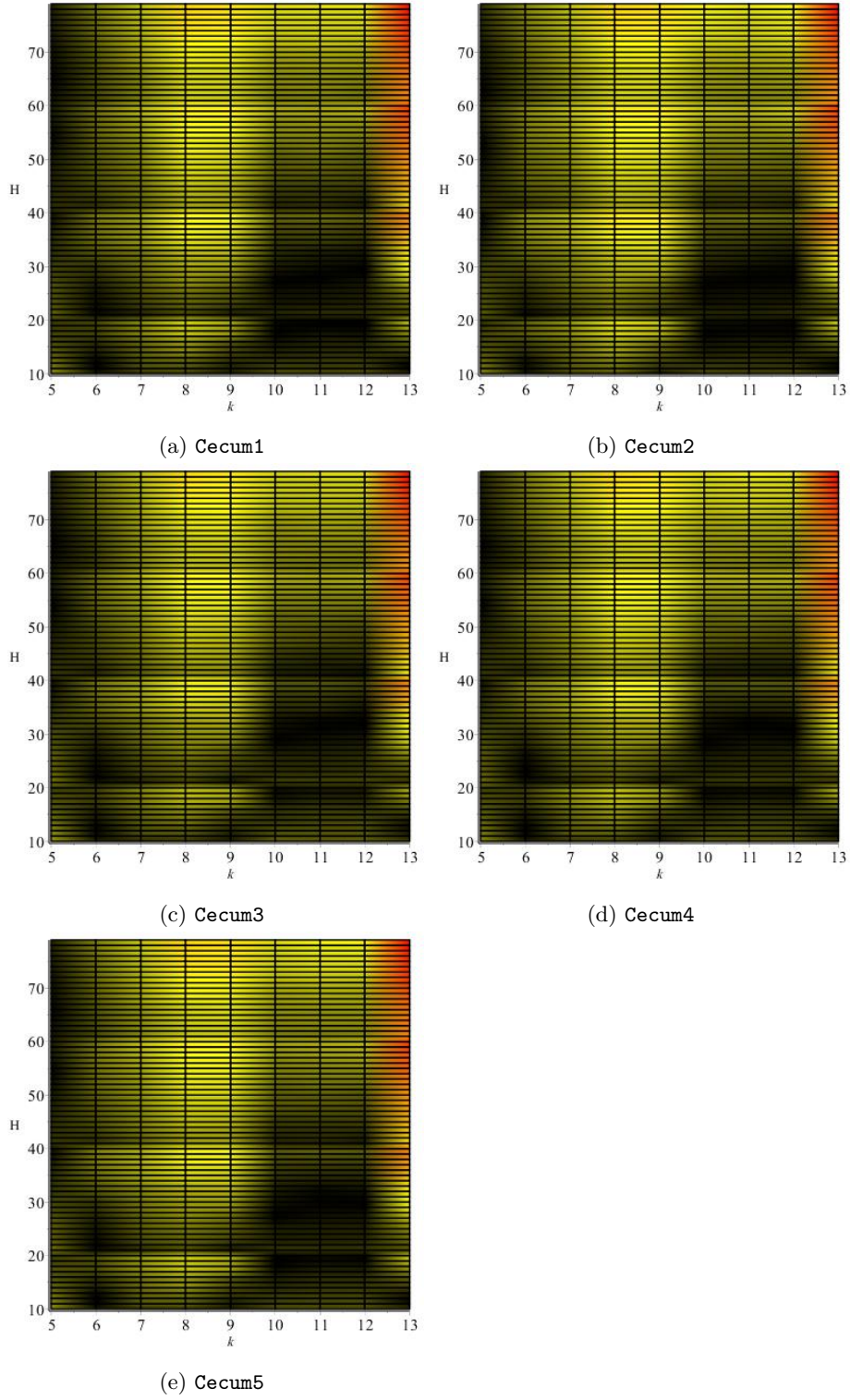
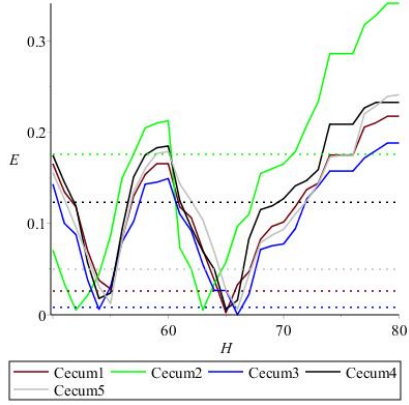
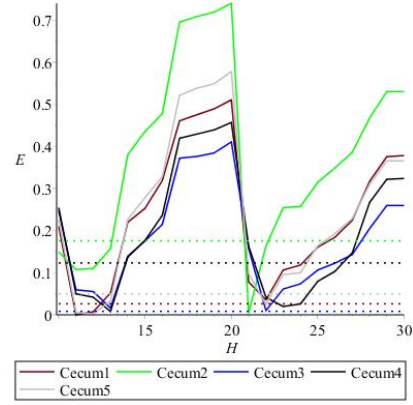


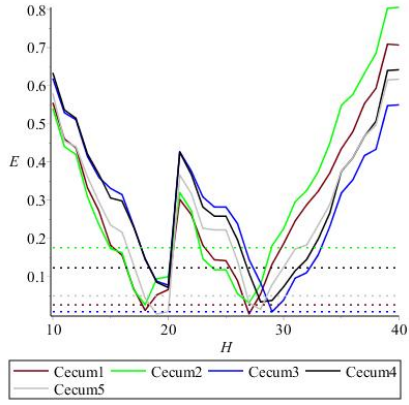
Figure 2: Heat maps of the error of **MM** at a narrow array of k -mer sizes k and numbers of hash functions H at all 5 samples of **Cecum** DNA. Black signifies the smallest error, red the highest error, and yellow medium error relative to black and red.



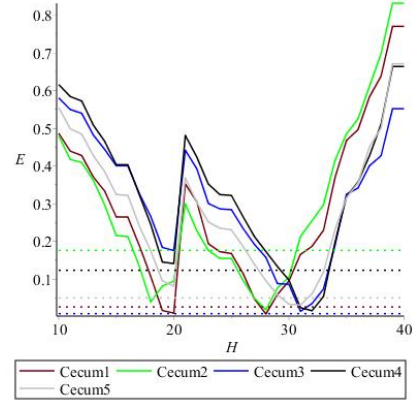
(a) $k=5$



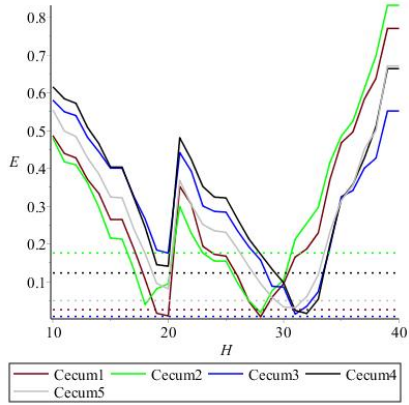
(b) $k=6$



(c) $k=10$



(d) $k=11$



(e) $k=12$

Figure 3: Plots with hash sizes H on the x -axis and error E_p between the test sample and the gold standard in percent ($0.5 = 50\%$) as defined in Eq. 1 on the y -axis, each plot showing results for a different k -mer size $k = 5, 8, 12$. The lines in the plots signify the error at the given k and H of **MM**, while the dotted lines signify the error of **UCLUST** for each file. Each sample file is given a unique color.

1.1.2 $\mathbf{MM}_{\frac{1}{2}}$ precision tests

The tests of $\mathbf{MM}_{\frac{1}{2}}$ were performed completely parallel to those of \mathbf{MM} , but with lower H as reputedly $\mathbf{MM}_{\frac{1}{2}}$ only needs half as many hash functions as the minwise sketch¹. In Fig. 4 figures the test of a wide array of k and H .

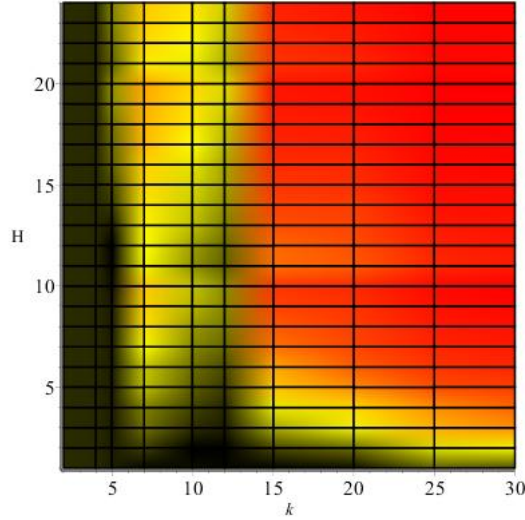


Figure 4: Heat map of the error of $\mathbf{MM}_{\frac{1}{2}}$ at a widely dispersed array of k and H , run on *Cecum1*. Black signifies the smallest error, red the highest error, and yellow medium error relative to black and red.

The large white plateau in Fig. 4 meant that $k > 15$ could be ignored, as the error at this area was very high. Similarly, $k < 5$ gave more error than $k = 5$ and could therefore be ignored. H could be reduced to $H < 15$, as the blackest spot seems around $k = 5$, $H = 12$.

New tests were performed with these limitations. The results can be seen in Fig. 5. All five plot are almost completely identical, showing that mostly at $H = 1$ and $H = 2$, the precision is highest. The only exception was at $k = 5$, where there seems to be a lower error along all H . An additional analysis showed that the minimum error in each heat map of Fig. 5 was always at $k = 8$, $H = 1$. This result was but a stroke of luck, since a single hash function would theoretically cause the error of the $\mathbf{MM}_{\frac{1}{2}}$ sketch to be too high to be applicable for proper comparison (see Eq. ??).

For further analysis of the k of interest, the errors of each file at $k = 5$, $k = 8$, and $k = 12$ were plotted, as seen in Fig. 6. Fig. 6(a-c) all have maximum errors that surpass 100%, even reaching as high as 700% error. Only in Fig. 6(a) a cuspidate spike was present at $H = 12$ with an average error of 20%, around the double of *UCLUST* average error.

¹see Section ??

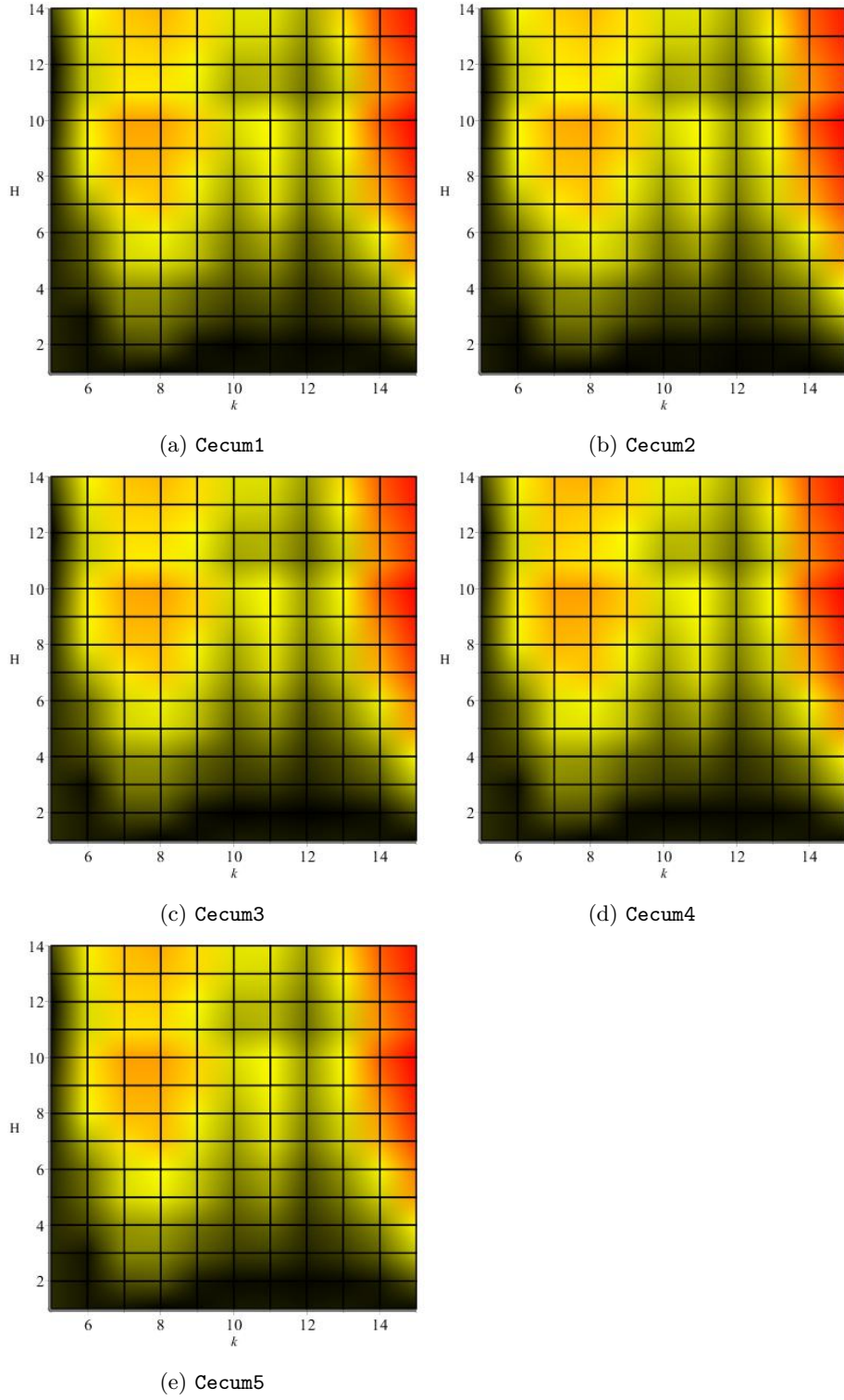


Figure 5: Heat maps of the error of $\text{MM}_{\frac{1}{2}}$ at a narrow array of k and H at all 5 samples of Cecum DNA. Black signifies the smallest error, red the highest error, and yellow medium error relative to black and red.

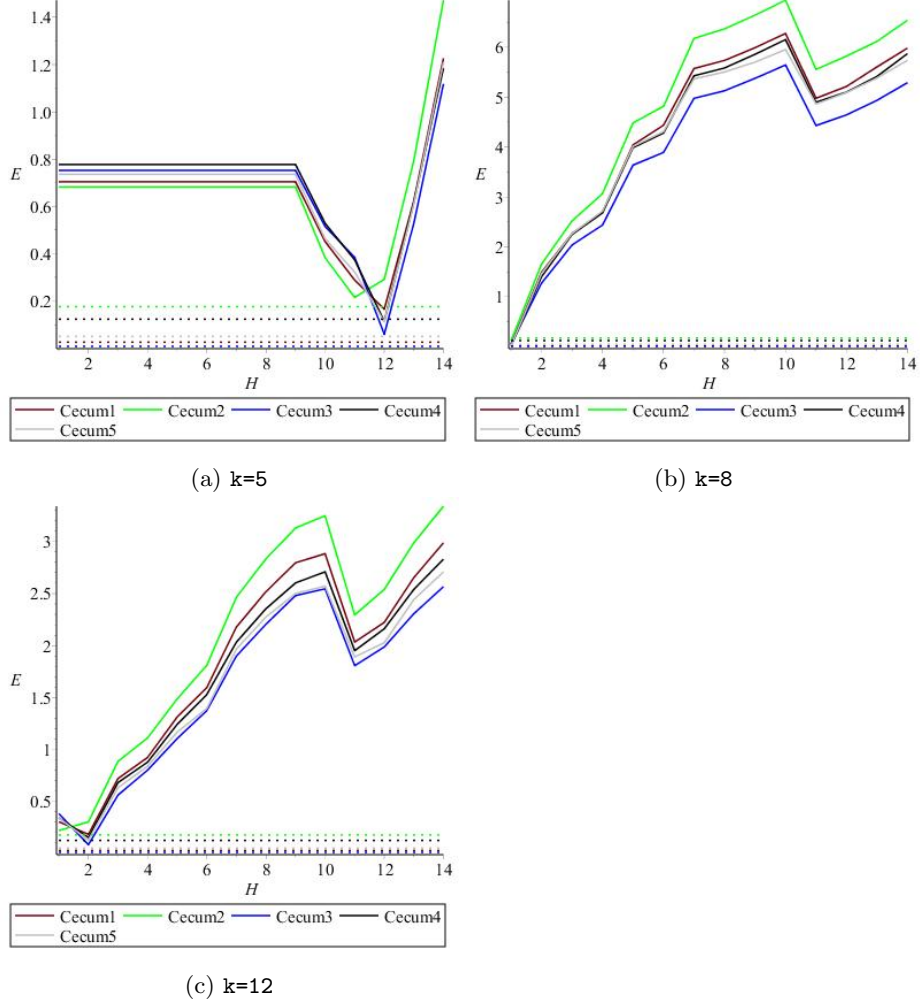


Figure 6: Plots with hash sizes H on the x -axis and error E_p between the test sample and the gold standard in percent ($0.5 = 50\%$) as defined in Eq. 1 on the y -axis, each plot showing results for a different k -mer size $k = 5, 8, 12$. The lines in the plots signify the error at the given k and H of $\text{MM}_{\frac{1}{2}}$, while the dotted lines signify the error of **UCLUST** for each file. Each sample file is given a unique color.

1.1.3 Comparison

To determine the precision of **MM**, $\text{MM}_{\frac{1}{2}}$ and **UCLUST**, Fig. 6 and Fig. 3 were perfect candidates. One of the most notable differences between the two graphs was the structure of the results for **MM**, which took an almost "W" like form. The reason for this is from how the sketches of two sequences are compared in Eq. ???. In contrast to Eq. ??, **MM** takes advantage of the strong points of both the max- and minwise hashing by finding the intersection of jaccards. This means that if the maxwise hashing is most precise at $H = 12$, and minwise is most precise at $H = 23$, the spikes will appear at these two spots.

Comparing the two, **MM** was quite simply much more precise than $\mathbf{MM}_{\frac{1}{2}}$, by very large margin. In fact, at the most precise, $\mathbf{MM}_{\frac{1}{2}}$ at $k = 5$ was still not as good as the worst **MM** was at $k = 5$ in our tests. In fact, **MM** turned out to be even more precise than **UCLUST** at some very specific parameter settings of k and H .

The extreme imprecision of $\mathbf{MM}_{\frac{1}{2}}$ was in fact so high, that it proved almost nonfunctional for practical purposes, if precision in any way was considered of any importance. **MM** however proved that it could compete with **UCLUST** in terms of precision. Therefore, **MM** was considered a good candidate for an alternative to **UCLUST**. What remained was to determine whether it could compare speed-wise.

1.2 Speed tests

For testing the speed, the MapReduce Framework using Pig Scripts were taken in use for **MM** and $\mathbf{MM}_{\frac{1}{2}}$. As only the 32 bit version of **UCLUST** was available, there was a memory limit to the size of the operation. For this reason, some of the sample files could not be finished by **UCLUST**; this occurrence will be noted as Mem. Exed.

To achieve the highest speed without loss of too much precision, the precision tests were used to determine the optimal parameters. **MM** was run at $k = 6$ and $H = 13$, which as seen in Fig. 3(b) is comparable to **UCLUST** in terms of precision. $\mathbf{MM}_{\frac{1}{2}}$ was run with $k = 5$ and $H = 12$.

The results of the test can be seen in Table 1. As expected, $\mathbf{MM}_{\frac{1}{2}}$ was consequently faster than **MM**. Theoretically, $\mathbf{MM}_{\frac{1}{2}}$ should be twice as fast as **MM**. This turned out true in all cases except for the runtime of **Actino1mio** and **Actino500K**. Here $\mathbf{MM}_{\frac{1}{2}}$ was 10 times faster than **MM**, caused by the fact that fewer clusters were produced by $\mathbf{MM}_{\frac{1}{2}}$. Therefore the outer loop of the $\mathbf{MM}_{\frac{1}{2}}$ greedy algorithm iterated less times than the outer loop of **MM**. Such a difference in speed was not predicted, but says more about the imprecision of $\mathbf{MM}_{\frac{1}{2}}$ than the algorithms' relative speed.

The reason why **UCLUST** was faster in both **Silva50K** and **Actino50K** laid in that the MapReduce Framework had a long startup time². If we had used a single node version of **MM** and $\mathbf{MM}_{\frac{1}{2}}$, we could probably have achieved much higher speeds at samples sizes of 50000 and less, since it would have no startup time. The MapReduce Framework would therefore first be advantageous to a single node solution using input files with 100.000 sequences or more.

Overall, there was a duality of speed between the two sample sets. In the **Actino** sample set, **UCLUST** consistently ran faster than both **MM** and $\mathbf{MM}_{\frac{1}{2}}$, with an increasing difference in speed over sample size. In the **Silva** samples, both **MM** and $\mathbf{MM}_{\frac{1}{2}}$ ran faster than **UCLUST** at sample size higher than **Silva50K**. In fact, **MM** was almost twice as fast as **UCLUST** on the run of **Silva500K**, and

²see section about MapReduce

the difference in speed increased with sample size.

While the frequency of cases where **MM** and $\mathbf{MM}^{\frac{1}{2}}$ are faster than **UCLUST** remains unknown, occasionally **MM** and $\mathbf{MM}^{\frac{1}{2}}$ trumps **UCLUST**. In choosing between $\mathbf{MM}^{\frac{1}{2}}$ and **MM**, a strong case could be made for **MM**. $\mathbf{MM}^{\frac{1}{2}}$ may have been double as quick as **MM**, but this at the cost of precision. So much precision, that one goes from statistically useless using $\mathbf{MM}^{\frac{1}{2}}$ to competing with **UCLUST** using **MM**. And since **MM** was also quicker than **UCLUST** in some cases, the results suggest that it could serve as a good alternative to **UCLUST**.

Sample	runtime of sample in s			% of UCLUST	
	UCLUST	MM	$\mathbf{MM}^{\frac{1}{2}}$	MM	$\mathbf{MM}^{\frac{1}{2}}$
Actino50K	2.0	18.3	13.6	915%	680%
Actino100K	5.0	23.2	18.7	464%	374%
Actino200K	9.0	53.4	28.8	593%	320%
Actino500K	14.0	208.4	58.3	1488%	416%
Actino1mio	Mem. Excd.	1033.8	103.3		
Silva50K	16.0	18.3	13.6	114%	85%
Silva100K	33.0	28.3	18.3	86%	55%
Silva200K	70.0	48.6	33.5	69%	48%
Silva500K	208.0	133.5	68.3	64%	33%
Silva1mio	Mem. Excd.	273.9	128.5		

Table 1: Comparison of clustering speed of three algorithms on 10 samples. **UCLUST** is run with `cluster_fast` option, **MM** with k -mer size $k = 6$ and $H = 13$ number of hash functions, $\mathbf{MM}^{\frac{1}{2}}$ with k -mer size $k = 5$ and $H = 12$ number of hash functions. The runtimes of all algorithms are given in seconds. The percentages are the relative speed of **MM** and $\mathbf{MM}^{\frac{1}{2}}$ to **UCLUST** in percent.