Spectral Jaccard Similarity: A new approach to estimating pairwise sequence alignments

Preprint · October 2019

DOI: 10.1101/800581

4 authors, including:

Tavor Baharav
Stanford University
4 PUBLICATIONS **34** CITATIONS
SEE PROFILE

Ilan Shomorony
University of California, Berkeley
36 PUBLICATIONS **447** CITATIONS
SEE PROFILE

Spectral Jaccard Similarity: A new approach to estimating pairwise sequence alignments

Tavor Baharav* [1], Govinda M. Kamath* [2], David N. Tse[1], and Ilan Shomorony[3]
[1] Department of Electrical Engineering, Stanford University.
[2] Microsoft Research New England, Cambridge, Massachusetts.
[3] Department of Electrical and Computer Engineering, University of Illinois, Urbana-Champaign.
tavorb@stanford.edu, gokamath@microsoft.com, dntse@stanford.edu, ilans@illinois.edu

10 October 2019

Abstract

A key step in many genomic analysis pipelines is the identification of regions of similarity between pairs of DNA sequencing reads. This task, known as pairwise sequence alignment, is a heavy computational burden, particularly in the context of third-generation long-read sequencing technologies, which produce noisy reads. This issue is commonly addressed via a two-step approach: first, we filter pairs of reads which are likely to have a large alignment, and then we perform computationally intensive alignment algorithms only on the selected pairs. The *Jaccard similarity* between the set of k-mers of each read can be shown to be a proxy for the alignment size, and is usually used as the filter. This strategy has the added benefit that the Jaccard similarities don't need to be computed exactly, and can instead be efficiently estimated through the use of *min-hashes*. This is done by hashing all k-mers of a read and computing the minimum hash value (the min-hash) for each read. For a randomly chosen hash function, the probability that the min-hashes are the same for two distinct reads is precisely their k-mer Jaccard similarity. Hence, one can estimate the Jaccard similarity by computing the fraction of min-hash collisions out of the set of hash functions considered.

However, when the k-mer distribution of the reads being considered is significantly non-uniform, Jaccard similarity is no longer a good proxy for the alignment size. In particular, genome-wide GC biases and the presence of common k-mers increase the probability of a min-hash collision, thus biasing the estimate of alignment size provided by the Jaccard similarity. In this work, we introduce a min-hash-based approach for estimating alignment sizes called *Spectral Jaccard Similarity* which naturally accounts for an uneven k-mer distribution in the reads being compared. The Spectral Jaccard Similarity is computed by considering a min-hash collision matrix (where rows correspond to pairs of reads and columns correspond to different hash functions), removing an offset, and performing a *singular value decomposition*. The leading left singular vector provides the Spectral Jaccard Similarity for each pair of reads. In addition, we develop an approximation to the Spectral Jaccard Similarity that can be computed with a single matrix-vector product, instead of a full singular value decomposition.

We demonstrate improvements in AUC of the Spectral Jaccard Similarity based filters over Jaccard Similarity based filters on 40 datasets of PacBio reads from the NCTC collection. The code is available at https://github.com/TavorB/spectral_jaccard_similarity.

*Contributed equally and listed alphabetically

1 Introduction

The advent of long-read sequencers such as PacBio and Oxford Nanopore have made the goal of obtaining gold-standard genome assemblies a reality. Unlike short read technologies, which provide reads of length 100-200 bp with an error rate of 1%, chiefly substitution errors, long read technologies provide reads of lengths in the tens of thousands with a nominal error rate of 13-15$\%$, consisting mostly of insertions and deletions [Weirather et al., 2017]. While the long reads make resolving repeated sequences easier, the higher error rates make the computational tasks required for assembly significantly more challenging.

Genome assembly is usually performed based on one of two main approaches: *de-novo* assembly, where one attempts to assemble a new genome "from scratch" using only the reads obtained, and *reference-based* assembly, where one assembles the reads using a pre-assembled genome of a related organism. Alignment is an integral part of most pipelines in either approach, and is often the most time consuming step (see Fig. 18). In both settings naive dynamic programming based alignment [Needleman and Wunsch, 1970, Smith and Waterman, 1981, Myers, 1986] is impractical due to its quadratic time complexity.

In reference-based assembly pipelines [Vaser et al., 2017, Wick, 2019], where one has a reference of a related organism, the first step usually consists of aligning all reads to the reference; i.e., read-to-reference alignment. For n reads each of length L and a reference of length G, the time complexity of aligning all reads to the reference via dynamic programming is $O(nLG)$, which is impractical in settings where there are $n \sim 10^5$ reads, each of length $L \sim 10^4$, with G on the order of 10^7-10^{11} depending on the organism.

Similarly, the first step in most de-novo assembly pipelines [Chin et al., 2013, Berlin et al., 2015, Li, 2016, Chin et al., 2016, Koren et al., 2017, Kamath et al., 2017] is the pairwise alignment of all reads, which is computationally very costly. For n reads each of length L, the time complexity of aligning all pairs of reads would be $O(n^2L^2)$. Even for bacterial genome datasets where the number of reads obtained is on the order of $n \sim 10^5$, with $L \sim 10^4$, this is impractical. For most of the sequel, we discuss alignment in the context of pairwise read alignment. However, our ideas can be adapted to the read-to-reference alignment paradigm.

One key observation that helps alleviate this computational burden is that in practice one only cares about alignment between reads when there is a significant overlap. Further, as shown in Figure 1, in a typical dataset, more than 99.99% of pairs of reads do not have a significant overlap. Hence, most practical read aligners follow a *seed-and-extend* paradigm. The "seeding" step typically involves identifying pairs of reads that share many k-mers (length-k contiguous substrings). This step can be understood as a way to "filter" the set of read pairs in order to select those that share a reasonable number of k-mers and are thus likely to have a significant overlap [Myers, 2014, Berlin et al., 2015, Li, 2016, 2018]. Once these "candidate pairs" (which are generally many orders of magnitude smaller than the number of read pairs) are obtained, computationally expensive dynamic-programming-based algorithms are used to obtain detailed alignment maps.

The idea of using the number of shared k-mers as a metric for filtering pairs of reads is equivalent to viewing the *Jaccard similarity* between the set of k-mers of each read as a proxy for the alignment size. Under standard implementations where computing set unions and set intersections has a linear time complexity in the sizes of the sets, this filtering step has a time complexity of $O(n^2L)$ for pairwise read alignment. Recently Jaccard similarity has been used in a variety of applications such as genome skimming [Denver et al., 2016], and in new methods to compare whole genomes and study taxonomic diversity in the microbiome [Ondov et al., 2016, Sarmashghi et al., 2019].

One very attractive property of Jaccard Similarity in the context of filtering pairs of reads is that this metric is amenable to efficient estimation through the use of *min-hashes*. This is done by hashing all k-mers on a read (the total number of k-mers in a length-L read is $L - k + 1$) and computing the minimum hash value (the min-hash) for each read. For a randomly chosen hash function, the collision probability for the min-hashes of two reads is precisely their Jaccard similarity. Hence, one can estimate the Jaccard similarity by computing the fraction of min-hash collisions out of the set of hash functions considered. For pairwise read alignment, if one uses H hashes to estimate the Jaccard similarity, this requires $O(nLH)$ to compute the

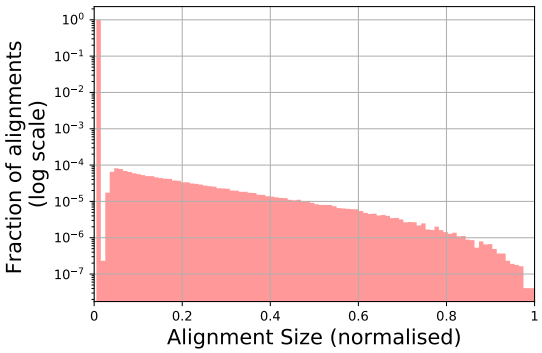


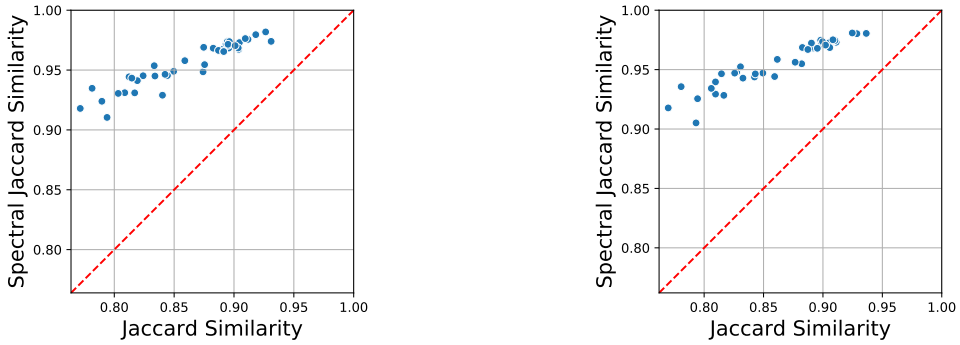
Figure 1: A histogram of fraction of shared sequence detected by Daligner Myers [2014] in reads of *E. coli* K-12 dataset of Pacific Biosciences Inc. [2013]. We note that more than 99.9% of pairs of reads have no alignment between them. We also note that practical aligners are not able to capture small overlaps, which are hard to distinguish from spurious alignments generated by noise, creating the "notch" in the histogram.

min-hashes and $O(n^2H)$ to compute the collisions giving us a time complexity of $O(n^2H)$ for the filtering step as generally, for regimes of interest, $n \gg L$.

This approach provides significant computational savings and is particularly effective when the genome where the reads come from is close to a random genome; i.e., a genome where every k-mer is equally likely to appear on a read. However, when the k-mer distribution of the reads being considered is significantly non-uniform, the Jaccard similarity is no longer a good proxy for the alignment size. In particular, genome-wide GC biases and the presence of common k-mers increase the probability of a min-hash collision, thus biasing the estimate of alignment size provided by the Jaccard similarity. In this work, we introduce a min-hash-based approach for estimating alignment sizes called *Spectral Jaccard Similarity*, which naturally accounts for an uneven k-mer distribution in the reads being compared. The Spectral Jaccard Similarity is computed by considering a min-hash collision matrix (where rows correspond to pairs of reads and columns correspond to different hash functions), removing an offset and performing a *singular value decomposition*. The leading left singular vector provides the Spectral Jaccard Similarity for each pair of reads, while the corresponding right singular vector can be understood as a measure of the "unreliability" of each hash function. Intuitively, a hash function that assigns low values to common k-mers is more unreliable for estimating alignment sizes, since it is more likely to create spurious min-hash collisions. Implicitly, this approach leads to a kind of weighted Jaccard similarity, where the weight of different hash functions is learned from the dataset.

Experiments on PacBio long-read sequencing data from several bacterial genomes, spanning a variety of k-mer distributions, show that the Spectral Jaccard Similarity is significantly more correlated with alignment size than the standard Jaccard Similarity. When used as a metric to filter out pairs of reads that are unlikely to have a large alignment, Spectral Jaccard Similarity outperforms Jaccard Similarity on standard classification performance metrics. As an example, when applied to filtering pairs of reads which have an overlap of at least 30%, the area under the ROC curve (AUC) obtained by Spectral Jaccard Similarity filtering was consistently higher than the AUC obtained for Jaccard similarity on 40 datasets of the NCTC collection of Parkhill et al, as shown in Figure 2. These results are obtained using $k = 7$, which as an appropriate choice for the PacBio error rates [Myers, 2014].

This manuscript is organized as follows: in Section 2, we present a brief review of Jaccard similarity and its application to seed-and-extend algorithms for pairwise read alignment; in Section 3 we present the basis for Spectral Jaccard similarity and in Section 4 we show results on real and simulated datasets. We then conclude with a discussion in Section 5.



(a) (b)

Figure 2: SJS has uniformly higher area under the ROC curve for experiments on 40 Pacbio bacterial datasets from the NCTC library [Parkhill et al]. In these experiments, Spectral Jaccard Similarity and Jaccard Similarity were used to filter pairs of reads with an overlap of at least 30%. SJS used was with 1000 hash functions, whil Jaccard similarity was computed exactly. (a) shows the AUC values using Daligner alignments as ground truth. (b) shows the same results using Minimap2 alignments as ground truth.

2 Jaccard Similarity

In general terms, the Jaccard similarity (JS) is a similarity metric between sets. For two sets A and B, the Jaccard similarity between them, $\mathrm{JS}(A, B)$, is defined as the size of their intersection divided by the size of their union. This is a very convenient measure as it is bounded between 0 and 1, $\mathrm{JS}(A, B) = 0$ if and only if $A \cap B = \emptyset$, and $\mathrm{JS}(A, B) = 1$ if and only if $A = B$. It has gained recent interest in its applications to finding documents (or web-pages) that are very similar but not the same as each other and in plagiarism detection. We refer the interested reader to Leskovec et al. [2014, Chapter 3] for a detailed review of the topic.

The Jaccard Similarity was applied to the problem of pairwise read alignment in Berlin et al. [2015], by considering the sets of k-mers of each read. For a fixed parameter k, the k-mer Jaccard similarity between reads S_0 and S_1, is given by

$$\mathrm{JS}_k(S_0, S_1) = \frac{|\Gamma(S_0) \cap \Gamma(S_1)|}{|\Gamma(S_0) \cup \Gamma(S_1)|}, \tag{1}$$

where $\Gamma(S_i)$ is defined as the set of k-mers for read S_i. This is the same as k-shingle Jaccard similarity in the data mining literature [Manber et al., 1994, Broder, 1997]. In this case, JS can be viewed as a proxy for the size of the overlap (if any) between reads S_0 and S_1.

For instance, consider length-L reads S_0 and S_1 with an overlap of size $\alpha_{0,1}L$, for $0 \leq \alpha_{0,1} \leq 1$, as illustrated in Fig. 3, and let $p_{0,1}$ be the fraction of overlap; i.e.,

$$p_{0,1} = \frac{\alpha_{0,1}}{2 - \alpha_{0,1}}. \tag{2}$$

If not many k-mers are shared by the non-overlapping parts of S_0 and S_1, we have that $\mathrm{JS}_k(S_0, S_1) \approx p_{0,1}$ making this a useful metric to filter pairs of reads with a high overlap. Notice that this approach is in a sense robust to errors in the reads. If we assume that each base is independently corrupted by noise (substitution, insertion or deletion) with probability z, then a k-mer is not corrupted with probability $(1 - z)^k$. Ignoring the unlikely event of collision of an erroneous k-mer with some other k-mer, $\mathrm{JS}_k(S_0, S_1)$ is approximated as

$$\mathrm{JS}_k(S_0, S_1) \approx \frac{\alpha_{0,1}(1 - z)^k}{2 - \alpha_{0,1}(1 - z)^k} \approx \frac{\alpha_{0,1}(1 - z)^k}{2}, \tag{3}$$

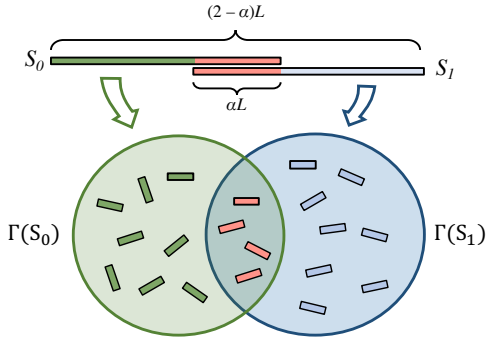


Figure 3: The k-mer Jaccard Similarity can be seen as a proxy for the alignment size.

where the last approximation holds when $(1-z)^k$ is small. Therefore, the k-mer Jaccard Similarity is intuitively still a good proxy for the overlap size in the presence of errors. The parameter k should be large enough to guarantee that not too many spurious k-mer collisions occur, but small enough so that a reasonable number of k-mers per read are not corrupted by noise [Myers, 2014, Berlin et al., 2015, Li, 2016]. For a relative noise rate of 30% (which results from both reads having error rates of around 15%), Myers [2014] argues that $k = 7$ achieves the optimal trade off. In the remainder of the manuscript, we utilize $k = 7$.

Computing the Jaccard Similarity between two reads of length L would take $O(L)$ time. Hence computing this metric for all pairs of reads would take $O(n^2L)$ time, which is quite expensive. An attractive feature of the JS metric is that it can be efficiently estimated. A probabilistic approach for estimating JS through the use of *min-hashes* was proposed in Broder et al. [2000]. In essence, one takes a random hash function h, hashes all k-mers in a read S_i and picks the minimum hash value. Define

$$h(\Gamma(S_i)) := \min\{h(x) : x \in \Gamma(S_i)\},$$

for some hash function h and read S_i. Then we observe that, for a randomly chosen hash function h,

$$\Pr\left[h(\Gamma(S_0)) = h(\Gamma(S_1))\right] = \mathrm{JS}_k(S_0, S_1), \tag{4}$$

since h is equally likely to have any of the k-mers on both reads as its minimizer. This means that we can use a random hash function to get an unbiased estimate of the Jaccard similarity between two strings. More precisely, if we sample random hash functions $h_1, h_2, ..., h_H$, we can estimate the Jaccard Similarity as

$$\frac{1}{H}\sum_{i=1}^{H} \mathbb{1}\left\{h_i(\Gamma(S_0)) = h_i(\Gamma(S_1))\right\} \overset{(a)}{\approx} \mathrm{JS}_k(S_0, S_1) \overset{(b)}{\approx} \frac{\alpha_{0,1}(1-z)^k}{2}. \tag{5}$$

Hence, by choosing H moderately large, one should be able to accurately estimate $\mathrm{JS}_k(S_0, S_1)$, which provides a proxy for the alignment size. With H hashes, one would take $O(nLH)$ time to compute the hashes and $O(n^2H)$ time to compute collisions, which is $O(n^2H)$ time in regimes of interest where $n \gg L$.

2.1 Drawbacks of Jaccard Similarity

The key assumption that drives the approximation in equation (5b) is that that all k-mers are roughly equally likely to occur in the reads. On real datasets however, k-mer distributions are far from uniform, as illustrated in Fig. 4. An uneven distribution of the k-mers throughout a genome increases the likelihood that the non-overlapping parts of two reads S_0 and S_1 share k-mers. In this case, for a randomly drawn hash function h, (4) still holds, but Eq. (5b) no longer holds. In particular, if the hash function h is such that common k-mers are given low hash values, the min-hash *collision probability*, given by $\Pr\left[h(\Gamma(S_0)) = h(\Gamma(S_1))\right]$,

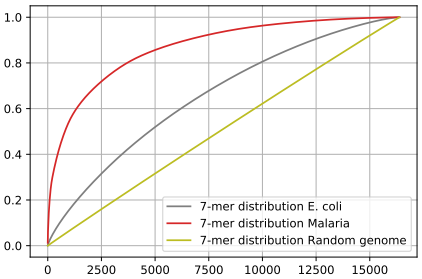


Figure 4: Cumulative distribution function (CDF) of the k-mer distributions for various genomes. For each genome, we sort the k-mers in decreasing order of frequency to help with visualization. We see that the distributions deviate significantly from a uniform distribution (dark yellow line).

can be significantly higher than the right-most expression in equation (5). For this reason, when the k-mer distribution throughout a genome is uneven, (5) yields a poor estimate for the read overlap size. This is illustrated in Fig. 6(a) for *E. coli* reads from [Pacific Biosciences Inc., 2013], where we show that Jaccard similarity is a poor predictor of alignment sizes.

One simple way to address this issue is to "mask" common k-mers [Myers, 2014, Koren et al., 2017], and then compute the Jaccard similarity on the remaining k-mers. However, these approaches are arbitrary and require the tuning of parameters that can in general depend on the distribution. Intuitively, they can be thought of as applying a hard threshold to determine which k-mer matches are due to noise, and which are actual signal. Our approach can be thought of as soft version of this thresholding, where we weight the importance of each min-hash collision differently.

3 Spectral Jaccard Similarity

We propose a new Jaccard-similarity-inspired approach to estimate the overlap between reads that avoids the need for hard thresholds for determining "common k-mers" or "bad hashes" and instead assigns soft penalties to individual hash functions according to how biased an estimator they are for alignment size.

Suppose reads S_0 and S_1 of length L have an overlap of size αL for some $0 \leq \alpha \leq 1$, and no other significant repeats across them. If there were no shared k-mers in the non-overlapping part of the reads, we would model the min-hash collision event for a random hash function h, as

$$\mathbb{1}\{h(\Gamma(S_0)) = h(\Gamma(S_1))\} \sim \text{Ber}(p_{0,1}), \tag{6}$$

where $p_{0,1} = \frac{\alpha}{2-\alpha}$ (this expression can be modified to account for errors as in (3)). However, when the distribution of k-mers is uneven, the min-hash collision probability is larger than $p_{0,1}$. Moreover, some hash functions are worse than others: if h assigns lower values to common k-mers, it is more likely to overestimate $p_{0,1}$. We model this effect by rewriting (6) as

$$\mathbb{1}\{h(\Gamma(S_0)) = h(\Gamma(S_1))\} \sim \text{Ber}(p_{0,1}) \vee \text{Ber}(q_h), \tag{7}$$

where $\vee$ is the boolean "or" operator and $q_h \in [0, 1]$ is a hash-specific parameter that can be intuitively understood as how unreliable h is due to common k-mers. Notice that the hash-specific noise term always leads to overestimation of $p_{0,1}$. We also emphasize that the q_h's are unknown. Therefore, we cannot directly estimate the $p_{i,j}$ and instead we need to jointly estimate all model parameters.

In order to perform this joint estimation, we define the *min-hash collision matrix* as follows. For a fixed reference read S_0, a list of target reads $S_1, S_2, ..., S_n$, and a list of hash functions $h_1, ..., h_H$, the (i, j)th entry

of the min-hash collision matrix is the binary indicator variable for whether there is a min-hash collision between S_0 and S_i when using hash function h_j; i.e., $\mathbb{1}\{h_j(\Gamma(S_0)) = h_j(\Gamma(S_i))\}$. Notice that $\mathrm{JS}_k(S_0, S_i)$ can be directly estimated from the min-hash collision matrix by computing the fraction of 1s in the ith row.

As it turns out, if we assume that the entries in the min-hash collision matrix were generated according to (7), we can jointly estimate the $p_{0,i}$'s and the q_{h_j}'s by performing a singular value decomposition (SVD) on an offset version of the min-hash collision matrix. This allows us to use efficient algorithms for computing the SVD in order to obtain estimates for the parameters $p_{0,i}$.

We refer to the parameters $p_{0,i}$ as the Spectral Jaccard Similarity (SJS) between S_0 and S_i. Intuitively, this value can be understood as a "discounted" version of $\mathrm{JS}_k(S_0, S_i)$, where we discount the contribution of common k-mers. It is important to point out, however, that p_i's are only implicitly defined as model parameters that are learned from the data, and no explicit formula for them exists. Next we describe the computation of the SJS in more detail.

3.1 Computing the Spectral Jaccard similarity

For a reference read S_0, we define the min-hash collision matrix as $A_0 \in \{0,1\}^{n \times H}$ where

$$A_0[i,j] = \mathbb{1}\{h_j(\Gamma(S_0)) = h_j(\Gamma(S_i))\} \sim \mathrm{Ber}(p_{0,i}) \vee \mathrm{Ber}(q_{h_j}). \tag{8}$$

For cleanliness of notation, we will write $p_{0,i} = p_i$ and $q_{h_j} = q_j$. Notice that both the p_i's and the q_j's depend on the choice of S_0, but we do not make that dependence explicit in the notation.

The key observation about our model is that, in expectation, the matrix A_0 defined in (8) is rank one after accounting for offset. More precisely, since $\mathbb{E}A_0[i,j] = p_i + q_j - p_i q_j = (1-p_i)(q_j - 1) + 1$ we have

$$\mathbb{E}A_0 - \mathbf{1}\mathbf{1}^\top = (\mathbf{1} - \mathbf{p})(\mathbf{q} - \mathbf{1})^\top. \tag{9}$$

We illustrate this point by comparing the sorted singular values of $A_i - \mathbf{1}\mathbf{1}^\top$ for the PacBio *E. coli* K-12 dataset. There is an order of magnitude gap in the difference between the first and second singular values, supporting the hypothesis that the min-hash collision matrix can be approximately modeled as a rank-one matrix, as shown in Figure 15. Focusing on read 0, we note that the fact that $A_0 - \mathbf{1}\mathbf{1}^\top$ is in expectation a rank-one matrix, allows us to estimate $\mathbf{p}$ and $\mathbf{q}$ through an SVD on $A_0 - \mathbf{1}\mathbf{1}^\top$. More precisely, if we let $\mathbf{u}$ and $\mathbf{v}$ be respectively the leading left and right singular vectors of $A_0 - \mathbf{1}\mathbf{1}^\top$, then we expect $\mathbf{u}$ to be approximately proportional to $(\mathbf{1} - \mathbf{p})$ and $\mathbf{v}$ to be approximately proportional to $(\mathbf{q} - \mathbf{1})$, up to flipping signs. We then normalize q_j's to be between 0 and 1, assuming that $\min_j q_j = 0$. For p_i's, we require a slightly more sophisticated normalization method, which we discuss in Section 5.1.

To illustrate the comparison between SJ and SJS we consider the example shown in Figure 5. The standard min-hash JS approach would estimate $\mathrm{JS}_k(S_0, S_i)$ to be the fraction of 1s in the ith row. We see that while rows 1 and 3 have the same estimated JS, they have different SJS values. This is because columns 2 and 5 are found to be noisier (i.e., worse hash functions), and so while rows 1 and 3 have two collisions each, a collision on column 1 is deemed more indicative of alignment, and thus row 3 has a higher SJS than row 1.

As it turns out, the estimates of p_i obtained via SVD are a much better proxy for the size of the alignment between reads S_0 and S_i than the standard Jaccard similarity, particularly when the k-mer distribution is uneven. To illustrate this point, we computed the SJS values p_i (from a min-hash collision matrix with $n = 1004$ and $H = 1000$) and the *exact* Jaccard Similarity $\mathrm{JS}_k(S_0, S_i)$ for the corresponding pairs of reads, for the *E. coli* PacBio dataset. As illustrated in Fig. 6, while the R^2 coefficient between (exact) JS and overlap size is only 0.18, for SJS the R^2 coefficient is 0.48.

While p_i parameters, or the SJS values, track the alignment between pairs of reads, we have not given a precise meaning to the q_js we recover. Intuitively, a large q_j means that the corresponding hash function is more likely to create a spurious min-hash collision, thus being a less reliable estimator for the alignment size;

	h_1	h_2	h_3	h_4	h_5	JS	SJS	q_1	q_2	q_3	q_4	q_5
S_1	0	1	0	0	1	0.4	0.198	0.187	0.504	0.054	0	0.813
S_2	0	0	0	0	0	0.0	0.0					
S_3	1	0	0	0	1	0.4	0.291					
S_4	0	1	0	0	1	0.4	0.198					
S_5	0	0	0	0	1	0.2	0.054					
S_6	1	1	1	0	1	0.8	0.709					
S_7	0	1	0	0	1	0.4	0.198					

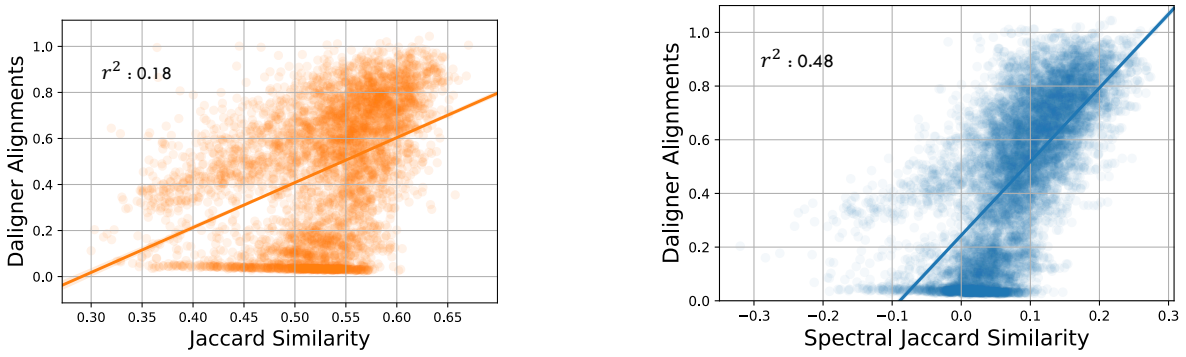
Figure 5: Example of comparison between JS and SJS on a small matrix. While the standard JS approach would assign the same value to rows 1 and 3, SJS takes into account the fact that columns 2 and 5 are seen as less reliable indicators of alignment.

we give a more in depth interpretation in Section 5.3. In Figures 10(a) and 10(b), we plot the frequency of the argmin k-mer for different hash functions, and and verify that large q_js correspond to hash functions whose argmin k-mers are common k-mers.

We conclude this section by noticing that, whenever most of the (S_0, S_i) pairs have no overlap, most of the p_i's are expected to be close to zero. When this holds true, a full SVD doesn't need to be computed. As we discuss in Section 5.2, in this case, the SVD can be approximated using a single matrix-vector product, allowing our method to be sped up significantly.

4 Results

To compare the performance of JS and SJS at estimating alignment sizes we focus on two PacBio datasets (an *E. coli* dataset [Pacific Biosciences Inc., 2013] and a *K. pneumoniae* (NCTC5047) dataset [Parkhill et al]) and consider the problem of identifying pairs of reads with an overlap of size at least θ. We compute exact JS values and compare them with SJS values computed based on 1000 hash functions (see Section 5.4 for results with different numbers of hash functions). We utilize Daligner [Myers, 2014] alignments as ground truth for the alignment sizes. In Figure 7, we plot ROC curves for different values of θ. We point out that using the Daligner outputs as ground truth is not ideal, since the tool itself utilizes an empirical Jaccard Similarity based filter to align reads; this choice of ground truth biases the result in favor of the conventional Jaccard similarity. Despite this we note that SJS performs significantly better than JS on all data sets tested. In addition, we obtain similar results when we use Minimap2 instead of Daligner to define the ground truth



(a) (b)

Figure 6: Linear regression fit to positive alignments found by Daligner to (a) Jaccard similarity between corresponding reads and (b) Spectral Jaccard similarity between the reads. Note that SJS provides a better fit.

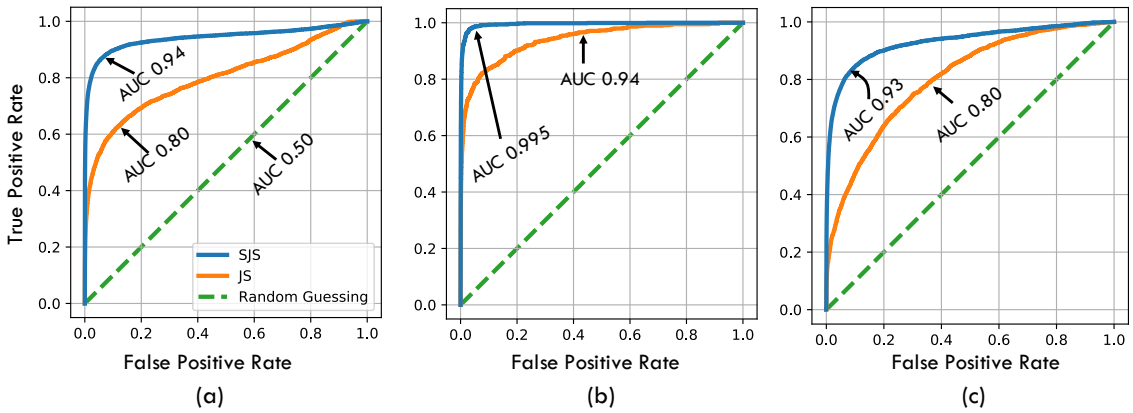


Figure 7: ROC curves across different PacBio datasets and different θ thresholds using Daligner ground truth and a 1000 hashes: (a) shows ROC of *E. coli* (K-12 from PacBio website) for alignment threshold $\theta = 0.3$, (b) shows ROC of *E. coli* for alignment threshold $\theta = 0.8$, and (c) shows ROC of *K. Pneumoniae* (NCTC5047) for alignment threshold $\theta = 0.8$. Figure 16 shows a similar plot with minimap2 ground truth.

as in Figures 2 and 14.

We note that the performance of both JS and SJS filters degrades as the k-mer distribution becomes skewed. This skew can be captured by the *collision probability* for a given k-mer distribution. We define the collision probability as the probability that a randomly selected read from the k-mer distribution of the dataset has a min-hash collision with a bag of L randomly sampled k-mers from the k-mer distribution of the data set. This can be computed in closed form, as we discuss in Appendix A. In Fig. 8(a) we plot the performance of SJS and JS as a function of the computed collision probability of a dataset for 40 datasets from the NCTC3000 project [Parkhill et al]. This shows the uniform improvement in performance afforded by SJS, in that for every dataset the SJS AUC is higher than the JS AUC. Further, it shows that as the k-mer distribution becomes more skewed, the degradation in performance suffered by SJS is smaller than that suffered by JS. We plot the ratio of the improvement of two AUCs over random guessing in Fig. 8(b). This shows that the improvement of SJS over JS is larger when the k-mer distribution is more skewed.

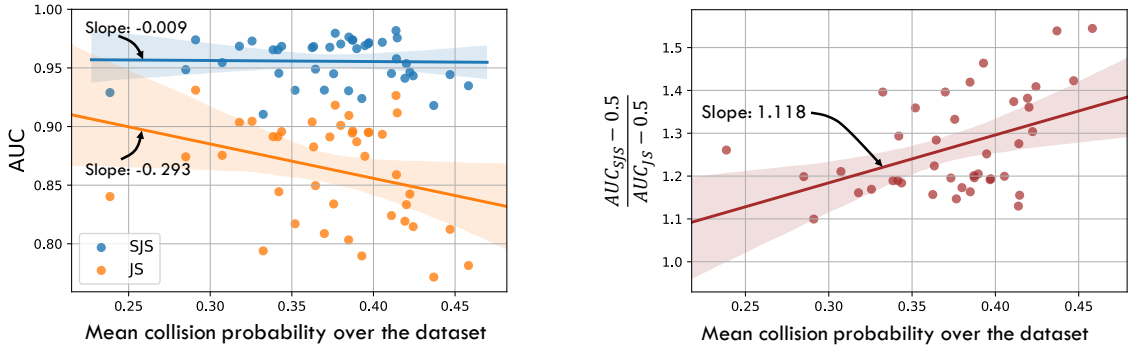


Figure 8: (a) The higher the min-hash collision probability is, the worse both methods perform, indicating a "harder" dataset. However, the performance of the SJS filter degrades less than that of the JS filter; (b) Ratio between the improvement of the SJS filter over random guessing and the improvement of the JS filter over random guessing, as a function of collision probability of the reads' k-mer distribution. Both plots using 1000 hashes.

5 Discussion

In this paper, we introduced the notion of Spectral Jaccard Similarity as an alternative to the standard k-mer Jaccard Similarity for estimating the overlap size between pairs of noisy, third-generation sequencing reads. SJS is a probabilistic approach that utilizes min-hash collisions as a way to estimate the size of the overlap between pairs of reads. However, unlike previous approaches, SJS attempts to learn how good different hash functions are at estimating overlap size for that *specific dataset*. In particular, when the k-mer distribution of the dataset in question is very uneven, the gain of SJS over JS is greater.

We conclude the paper by discussing four points: the first two are algorithmic aspects of SJS that, while not central to understanding the method, are important from an implementation standpoint. The third is another empirical validation of our model, and the last is the performance of SJS as we vary the number of hashes used. First, we discuss how we normalize, or calibrate, the p_is obtained for different reference reads (i.e., from the SVD of different matrices A_i and A_j) so that the SJS are comparable. Second, we show how the fact that the columns of the A_0 matrix are typically sparse can be exploited in order to approximate the SVD using a single matrix-vector multiplication, which can significantly speed up the computation of SJS. Third, we validate our earlier claim that q_js represent how bad a hash function is for the purpose of alignment. Finally, we examine the performance of SJS as function of the number of hashes used and show that it can match the performance of exact Jaccard similarity with around 150 hashes.

5.1 Calibrating SJS values from different reference reads

Given two min hash collision matrices A_0, $A_1 \in \{0,1\}^{n\times H}$ we have from (9) that $A_k - \mathbf{1}\mathbf{1}^\top \approx (\mathbf{1} - \mathbf{p}_k)(\mathbf{q}_k - \mathbf{1})^\top$ for $k \in \{0,1\}$. When one computes an SVD, the left and right singular vectors are scaled so that they have an ℓ_2 norm of 1. However for our purposes, normalizing these vectors by the ℓ_2 norm implies assuming that $\mathbf{p}_i$'s have similar norms for all reference reads i which need not be true. Notice that we could potentially have two constants $\gamma_0, \gamma_1 \neq 0$, $\gamma_0 \neq \gamma_1$ such that $A_k - \mathbf{1}\mathbf{1}^\top = \frac{1}{\gamma_k}(\mathbf{1} - \mathbf{p}_k)\gamma_k(\mathbf{q}_k - \mathbf{1})^\top$ for $k \in \{0,1\}$ With different γ_0 and γ_1, we see that we would not be able to compare values in $\mathbf{p}_0$ and $\mathbf{p}_1$, which leads to issues when one is trying to threshold all with a common threshold.

One approach to calibrating the $p_{0,i}$'s and $p_{1,i}$'s to make the left singular vectors of $A_0 - \mathbf{1}\mathbf{1}^\top$ and $A_1 - \mathbf{1}\mathbf{1}^\top$ comparable takes advantage of *calibration reads*. Calibration reads are fake reads, following the dataset's k-mer distribution, that are expected to have no overlap with the reference, thus making sure that the Spectral Jaccard Similarities computed on different runs of the SVD have the same "ground level", so that we can compare the SJS values between two pairs of reads coming from two different SVDs.

More precisely, our procedure involves including W calibration reads into the min-hash collision matrix. To create these calibration reads, we sample $L - k + 1$ k-mers from the k-mer distribution of the dataset, where L is the average read length of the dataset. Hence, each calibration read is essentially a "bag of $L - k + 1$ k-mers". Then we augment the matrix A_0 and consider instead a matrix $B_0 \in \{0,1\}^{(n+W)\times H}$ with the first n rows of B_0 being A_0 and the last W rows being the min-hash collision matrix computed for the W calibration reads. We then compute an SVD of $B_0 - \mathbf{1}\mathbf{1}^\top$ as

$$B_0 - \mathbf{1}\mathbf{1}^\top = (\mathbf{1} - \tilde{\mathbf{p}}_0)(\tilde{\mathbf{q}}_0 - \mathbf{1})^\top = \mathbf{u_0}\mathbf{v_0}^\top.$$

Using our random reads we define $u_0^{\text{med}} = \text{median}\,\{u_0[i], i \in [n, n+W-1]\}$. We now proceed to enforce that the median of $\{\tilde{p}_i, i \in [n, n+W-1]\}$ be 0 by defining the normalized SJS values as

$$p_{0,i} = 1 - \left|\frac{u_0[i]}{u_0^{\text{med}}}\right| \qquad \mathbf{q}_0[j] = 1 - \left|\frac{v_0[i]}{\max_{\ell\in[H]}|v_0[\ell]|}\right|$$

Remark 1. *We set the median alignment of the random reads to* 0 *rather than the minimum or maximum alignment because the variance in the extremal value was empirically found to be large.*

Remark 2. *In practice we use* $W = 5$*, but varying* W *does not seem to affect results significantly. Since* n *is typically large (in the thousands), these additional fake reads do not really change the result of the SVD.*

5.2 Approximation in the case where most p_i's are zero

Given a min-hash collision matrix $A \in \{0,1\}^{n \times H}$, define

$$\bar{p} = \frac{1}{n}\sum_{i=1}^{n} p_i, \qquad \bar{q}_j = \frac{1}{n}\sum_{i=1}^{n} A_0[i,j],$$

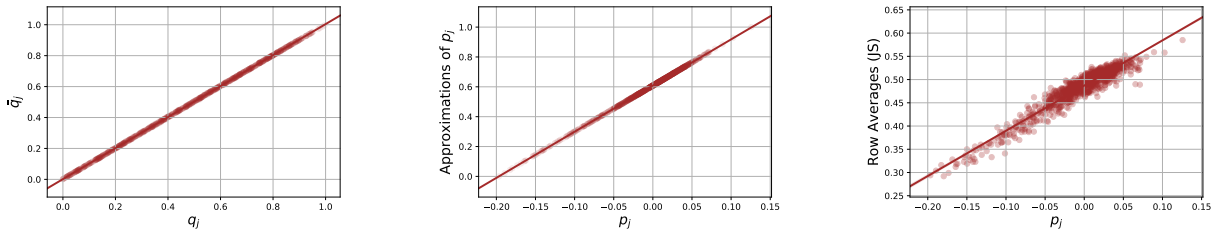
for $1 \leq j \leq H$; i.e., $\bar{p}$ is the average p_i value and $\bar{q}_j$ is the fraction of ones in column j. We note that, since $A_0[i,j] \sim \text{Ber}(p_i) \vee \text{Ber}(q_j)$, when most p_i's are zero, most of the entries in column j are distributed as $\text{Ber}(q_j)$. It follows that $\mathbb{E}[\bar{q}_j] = q_j + \bar{p} - \bar{p}q_j \approx q_j$ since $\bar{p} \approx 0$. This means that the leading right singular vector is approximately $\bar{\mathbf{q}} = [\bar{q}_1, ..., \bar{q}_H]^\top$. Since the rank-one approximation is

$$A_0 - \mathbf{1}\mathbf{1}^\top \approx (\mathbf{1} - \mathbf{p})(\mathbf{q} - \mathbf{1})^\top \approx (\mathbf{1} - \mathbf{p})(\bar{\mathbf{q}} - \mathbf{1})^\top,$$

by multiplying both sides by $(\bar{\mathbf{q}} - \mathbf{1})$, we obtain

$$\begin{aligned} (A_0 - \mathbf{1}\mathbf{1}^\top)(\bar{\mathbf{q}} - \mathbf{1}) &\approx \|\bar{\mathbf{q}} - \mathbf{1}\|_2^2(\mathbf{1} - \mathbf{p}) \\ \Rightarrow \mathbf{p} &\approx \mathbf{1} - \frac{1}{\|\bar{\mathbf{q}} - \mathbf{1}\|_2^2}(A_0 - \mathbf{1}\mathbf{1}^\top)(\bar{\mathbf{q}} - \mathbf{1}) \end{aligned} \tag{10}$$

which gives us a method to compute the Spectral Jaccard Similarity with a matrix-vector multiplication, rather than an SVD. This can intuitively be understood as follows. We wish to approximate the principal left singular vector of the matrix $A_0 - \mathbf{1}\mathbf{1}^\top$. We are, however given some side information; we are able to easily obtain a high-quality approximation of the principal right singular vector as $\bar{\mathbf{q}} - \mathbf{1}$. This allows us to effectively perform one step of the Power Iteration algorithm, as $(A_0 - \mathbf{1}\mathbf{1}^\top)(\bar{\mathbf{q}} - \mathbf{1})$, which after normalization, gives us a very good approximation of the principal left singular vector.



(a) (b) (c)

Figure 9: When most $p_i \approx 0$, it is possible to approximate the SVD by a simple matrix-vector multiplication as described in (10). In particular, we verify empirically that (a) $\bar{\mathbf{q}} \approx \mathbf{q}$ and that (b) the $\mathbf{p}$ obtained from (10) is nearly the same as the one computed by SVD. (c) If one instead tries to approximate $\mathbf{p}$ by considering row averages, the approximation is not as good.

In Figure 9(a), we show that for an *E. coli* dataset where most reads do not have any overlap, $\bar{\mathbf{q}}$ is very correlated with $\mathbf{q}$. In Figure 9(b), we show that the approximate SJS values computed using (10) are highly correlated with those computed through a full SVD.

We note that one could have have attempted to use row averages instead of column averages in this approximation procedure. However, this would correspond to computing the standard Jaccard Similarities.

Jaccard Similarity is not as well correlated with the Spectral Jaccard Similarity as we show in Figure 9(c). Further, we note that (10) implies that $p_i \underset{\sim}{\propto} [A(1-\bar{\mathbf{q}})]_i$, which is expanded as

$$p_i \underset{\sim}{\propto} \frac{1}{H}\left(\sum_{j=1}^{H} A_{i,j}\left(1-\frac{1}{n}\sum_{\ell=1}^{n} A_{\ell,j}\right)\right), \tag{11}$$

where $\underset{\sim}{\propto}$ indicates "monotonic function of". Since $\mathrm{JS}_k(S_0, S_i) \approx \frac{1}{H}\sum_{j=1}^{H} A_{i,j}$, our method can be understood as down-weighting the contribution of hash functions that yield many collisions.

We show that this approximation performance performs nearly as well as SJS in Figure 12.

5.3 Validating the Model

In Section 3, we proposed the model in (7) with the interpretation that q_j represented how likely were min-hash collisions given the hash function h_j. In this section, we empirically verify that claim.

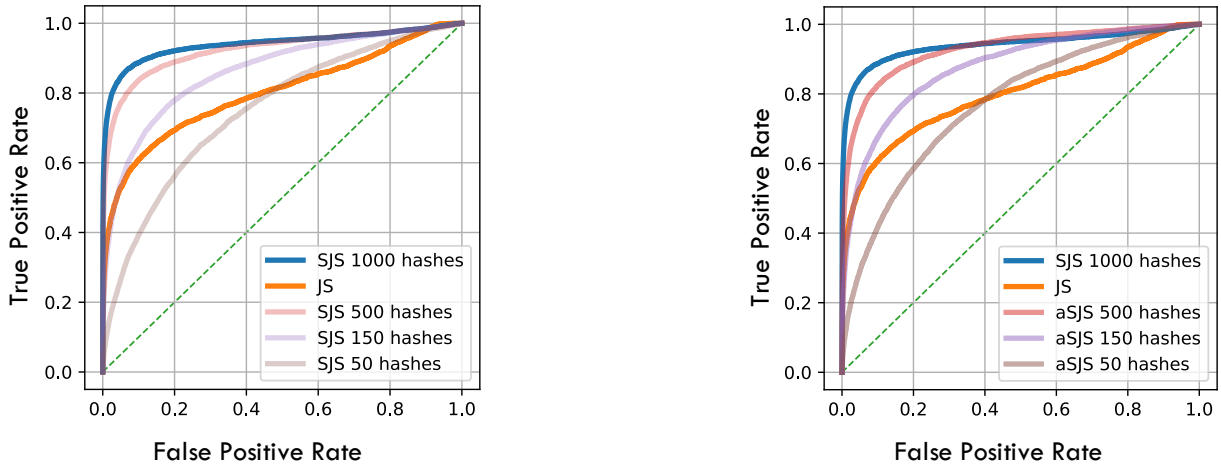
For a dataset with mean read length L, the collision probability of a read S on a hash function h is defined as the probability that $L-k+1$ samples from the k-mer distribution of the the dataset have the same min-hash on h as S. This can be computed in closed form as discussed in Appendix A. In Figure 10, we show the collision probability of a reference read on a hash function h_j as a function of our computed q_j for the *E. coli* and *K. pneumoneae* (NCTC5047) datasets. We see a very strong correlation between the computed collision probability and the q_j parameters, validating our model.

(a) (b)

Figure 10: For each hash function h_j, we compared the collision probability on hash h_j with the corresponding q_j for (a) the *E. coli* dataset and (b) the *K. pneumoniae* dataset (NCTC5047).

5.4 Trade-off between filter accuracy and number of hash functions

While throughout this manuscript we present results for SJS using 1000 hash functions, our method performs well even with a smaller number of hash functions. In Figure 11(a), we plot ROC curves for SJS and the SJS approximation (aSJS) described in Section 5.2 using different number of hashes in order to compare the performance of these filters on the *E. coli* K-12 dataset. We note that as few as 150 hashes are enough for SJS to dominate the exact JS based filter. The performance of the approximation is similar.



(a) (b)

Figure 11: ROC curves obtained by (a) SJS and (b) aSJS as we vary the number of hash functions used for $\theta = 0.3$.

References

K. Berlin, S. Koren, C.-S. Chin, J. P. Drake, J. M. Landolin, and A. M. Phillippy. Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nature biotechnology*, 33(6):623, 2015.

A. Z. Broder. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*, pages 21–29. IEEE, 1997.

A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3):630–659, 2000.

C.-S. Chin, D. H. Alexander, P. Marks, A. A. Klammer, J. Drake, C. Heiner, A. Clum, A. Copeland, J. Huddleston, E. E. Eichler, et al. Nonhybrid, finished microbial genome assemblies from long-read smrt sequencing data. *Nature methods*, 10(6):563, 2013.

C.-S. Chin, P. Peluso, F. J. Sedlazeck, M. Nattestad, G. T. Concepcion, A. Clum, C. Dunn, R. O'Malley, R. Figueroa-Balderas, A. Morales-Cruz, et al. Phased diploid genome assembly with single-molecule real-time sequencing. *Nature methods*, 13(12):1050, 2016.

D. R. Denver, A. M. Brown, D. K. Howe, A. B. Peetz, and I. A. Zasada. Genome skimming: a rapid approach to gaining diverse biological insights into multicellular pathogens. *PLoS Pathogens*, 12(8):e1005713, 2016.

G. M. Kamath, I. Shomorony, F. Xia, T. A. Courtade, and D. N. Tse. Hinge: long-read assembly achieves optimal repeat resolution. *Genome research*, 27(5):747–756, 2017.

S. Koren, B. P. Walenz, K. Berlin, J. R. Miller, N. H. Bergman, and A. M. Phillippy. Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome research*, 27(5): 722–736, 2017.

J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of massive datasets*. Cambridge university press, 2014.

H. Li. Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics*, 32(14):2103–2110, 2016.

H. Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 34(18):3094–3100, 2018.

U. Manber et al. Finding similar files in a large file system. In *Usenix Winter*, volume 94, pages 1–10, 1994.

E. W. Myers. Ano (nd) difference algorithm and its variations. *Algorithmica*, 1(1-4):251–266, 1986.

G. Myers. Efficient local alignment discovery amongst noisy long reads. In *International Workshop on Algorithms in Bioinformatics*, pages 52–67. Springer, 2014.

S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

B. D. Ondov, T. J. Treangen, P. Melsted, A. B. Mallonee, N. H. Bergman, S. Koren, and A. M. Phillippy. Mash: fast genome and metagenome distance estimation using minhash. *Genome biology*, 17(1):132, 2016.

Pacific Biosciences Inc. Pacbio e. coli dataset, 2013. URL `https://github.com/PacificBiosciences/DevNet/wiki/E.-coli-Bacterial-Assembly`.

J. Parkhill et al. National collection of type cultures (NCTC)- 3000. URL https://www.sanger.ac.uk/resources/downloads/bacteria/nctc/.

S. Sarmashghi, K. Bohmann, M. T. P. Gilbert, V. Bafna, and S. Mirarab. Skmer: assembly-free and alignment-free sample identification using genome skims. *Genome biology*, 20(1):34, 2019.

T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

R. Vaser, I. Sović, N. Nagarajan, and M. Šikić. Fast and accurate de novo genome assembly from long uncorrected reads. *Genome research*, 27(5):737–746, 2017.

J. L. Weirather, M. de Cesare, Y. Wang, P. Piazza, V. Sebastiano, X.-J. Wang, D. Buck, and K. F. Au. Comprehensive comparison of pacific biosciences and oxford nanopore technologies and their applications to transcriptome analysis. *F1000Research*, 6, 2017.

R. R. Wick. Rebaler - a reference-based long read assemblies of bacterial genomes, 2019. URL https://github.com/rrwick/Rebaler.

Appendices

A Collision Probability

We define collision probability for a length-L read S with a hash function h_j as the probability that a bag of random $L-k+1$ k-mers drawn from the dataset's k-mer distribution has the same min-hash on hash h_j as read S. For a dataset, we consider the average collision probability of all reads as the collision probability representative of the dataset.

We begin with some notation. First, note that each hash function can be thought of as inducing an ordering on the set of 4^k k-mers. We denote this permutation by $(\cdot)_j$ for hash h_j. Let $P^{(j)}_{(\ell)}$ be the probability (or the frequency) of the ℓ-th k-mer in the permutation defined by hash h_j (on the k-mer distribution) of the dataset. For example, $P^{(j)}_{(1)}$ denotes the probability of the k-mer hashed to the minimum value by hash h_j. Further, define partial sums

$$\zeta^{(j)}_{(\ell)} = \sum_{t=1}^{\ell} P^{(j)}_{(t)}.$$

In words, $\zeta^{(j)}_{(\ell)}$ are the probabilities of the first ℓ k-mers in the permutation induced by hash h_j (or equivalently, the probability of ℓ k-mers that hash h_j maps to the smallest values). Finally, we denote the smallest k-mer in the order induced by hash h_j that is also in read S_i as ℓ_i. Then the collision probability with the ith read as reference and hash function h_j is

$$\begin{aligned}
&\Pr(\text{random read collides with reference read } S_i \text{ on hash } h_j) \\
&\quad = \Pr(\text{random read has } k\text{-mer } (\ell_i)_j \text{ and does not have } k\text{-mers } (1)_j, \dots (i-1)_j) \\
&\quad = \Pr(\text{random read does not have } k\text{-mers} (1)_j, \dots (\ell_i - 1)_j) \\
&\qquad \times \Pr(\text{read has } k\text{-mer } (\ell_i)_j \mid \text{read does not have } k\text{-mers } (1)_j, \dots (\ell_i - 1)_j) \\
&\quad = \left(1 - \sum_{t=1}^{\ell_i - 1} P^{(j)}_{(t)}\right)^L \left(1 - \left[1 - \frac{P^{(j)}_{(\ell_i)}}{\sum_{t=\ell_i}^{4^k} P^{(j)}_{(t)}}\right]^L\right) \\
&\quad = \left(1 - \zeta^{(j)}_{(\ell_i - 1)}\right)^L \left(1 - \left[1 - \frac{P^{(j)}_{(\ell_i)}}{1 - \zeta^{(j)}_{(\ell_i - 1)}}\right]^L\right), \\
&\quad \approx \exp\left(-L\zeta^{(j)}_{(\ell_i - 1)}\right) \left(1 - \left[1 - \frac{P^{(j)}_{(\ell_i)}}{1 - \zeta^{(j)}_{(\ell_i - 1)}}\right]^L\right), \\
&\quad \overset{(a)}{\approx} \exp\left(-L\zeta^{(j)}_{(\ell_i - 1)}\right) \left(1 - \left[1 - P^{(j)}_{(\ell_i)}\right]^L\right), \\
&\quad \approx \exp\left(-L\zeta^{(j)}_{(\ell_i - 1)}\right) \left(1 - \exp\left(-LP^{(j)}_{(\ell_i)}\right)\right), \\
&\quad = \exp\left(-L\zeta^{(j)}_{(\ell_i - 1)}\right) - \exp\left(-L\zeta^{(j)}_{(\ell_i)}\right),
\end{aligned}$$

where the approximation in (a) follows in our setting where L and 4^k are of similar orders, and hence ℓ_i is

small and $\zeta^{(j)}_{(\ell_i-1)}$ is close to 0. From this we obtain

$$\begin{aligned}
&\Pr(\text{rand read collides with read } S_0)\\
&\quad= \sum_{m=0}^{4^k!} \Pr(\text{rand read collides with read } S_0 \text{ given hash permutation } (\cdot)_m)\\
&\qquad\times \Pr(\text{hash yields permutation } (\cdot)_m)\\
&\quad\approx \frac{1}{H}\sum_{j=1}^{H} \Pr(\text{rand read collides with read } S_0 \text{ given hash permutation } (\cdot)_j).
\end{aligned}$$

The first term above is exactly the probability we computed previously. This gives us an explicit way to (approximately) compute the collision probability.

B Details of experiments

We used the first 40 NCTC datasets of Parkhill et al (sorted numerically) for our empirical results. Since our main goal was to evaluate the performance of SJS (rather than fully re-align all the reads in these datasets), we pruned the datasets down so that each one contained 1000 reads. These reads were picked to match the distribution in Figure 1, such that each read has at least five significant overlaps. In essence, we went over the reads and added five reads that have large overlaps (according to Daligner [Myers, 2014]) with the read giving us a dataset where $\sim 0.5\%$ of read pairs have significant alignments. We then used Daligner and Minimap2 [Li, 2018] to generate alignments on these 1000 reads which we use as ground truths for AUC computations. The same is done for the *E. coli* K-12 dataset of Pacific Biosciences Inc. [2013] for the running example used throughout this manuscript. The code to reproduce our numerical results is available online at `https://github.com/TavorB/spectral_jaccard_similarity`.

C Additional Empirical Results

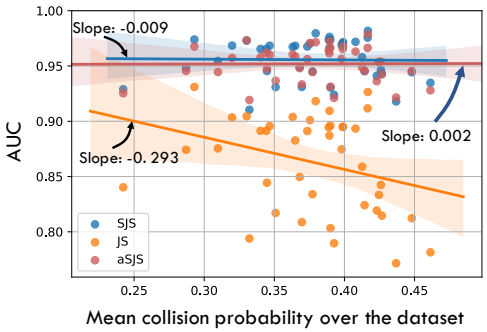
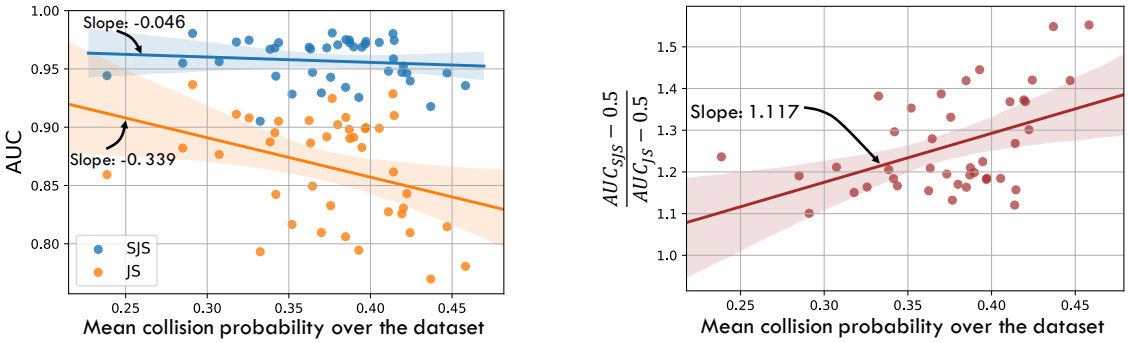


Figure 12: Redrawing Figure 8(a) with our power iteration approximation added in red as aSJS. As we can see, our one step of power iteration performs very well as an approximation for the SVD. While aSJS rarely outperforms SJS, it has most of the gain in predictive power while still being almost as computationally efficient as empirical JS, who's performance is slightly under that of JS.

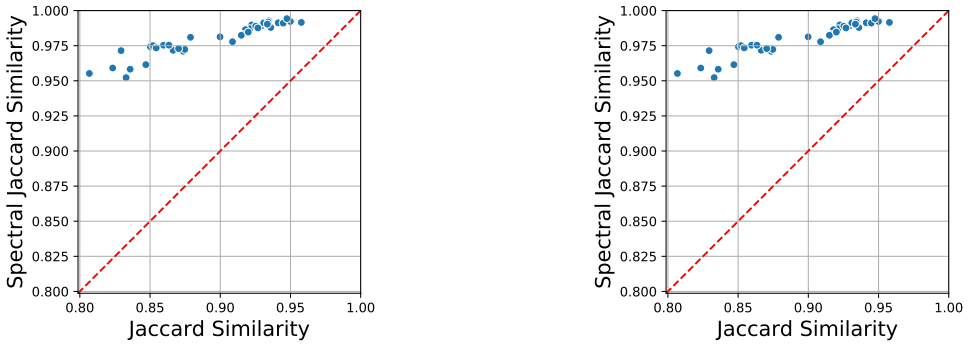
List of NCTC datasets used
NCTC74
NCTC86
NCTC92
NCTC129
NCTC204
NCTC235
NCTC418
NCTC1080
NCTC1936
NCTC2218
NCTC2366
NCTC2669
NCTC3046
NCTC3166
NCTC3168
NCTC3438
NCTC3750
NCTC3761
NCTC4001
NCTC4133
NCTC4136
NCTC4137
NCTC4163
NCTC4168
NCTC4169
NCTC4174
NCTC4199
NCTC4444
NCTC4450
NCTC4669
NCTC4671
NCTC4672
NCTC4673
NCTC4675
NCTC4725
NCTC4840
NCTC5046
NCTC5047
NCTC5050
NCTC5051
NCTC5052

Table 1: List of datasets used



(a) (b)

Figure 13: Analogue of Figure 8 using Minimap2 to provide the ground truth alignment sizes. The results are similar to those in Figure 8. Note that in (b) the slope is almost identical to what we obtained in the Daligner case in Fig. 8(b).



(a) (b)

Figure 14: Analogue of Figure 2 but for $\theta = 0.5$. (a) shows the ROCs using Daligner alignments as ground truth. (b) shows them using Minimap2 alignments as ground truth.

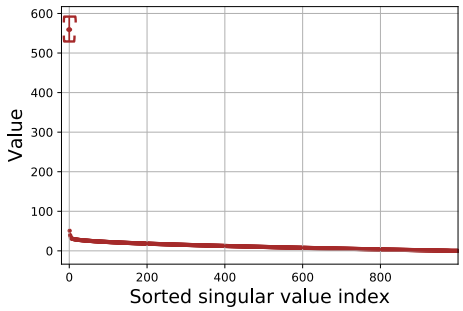
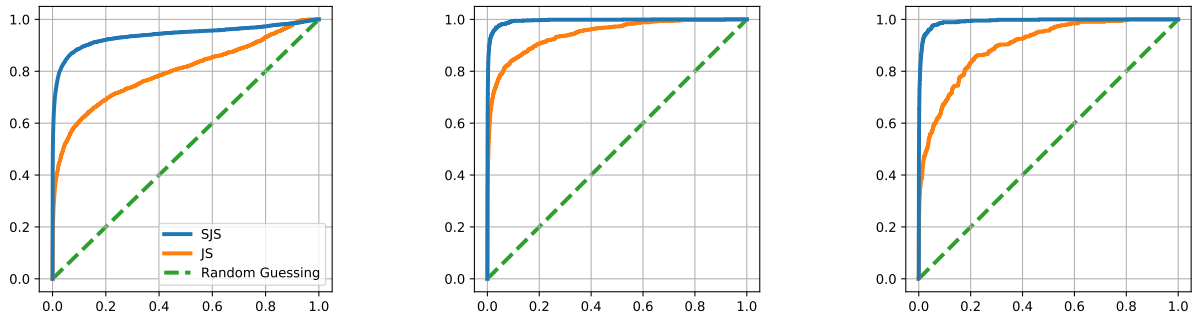
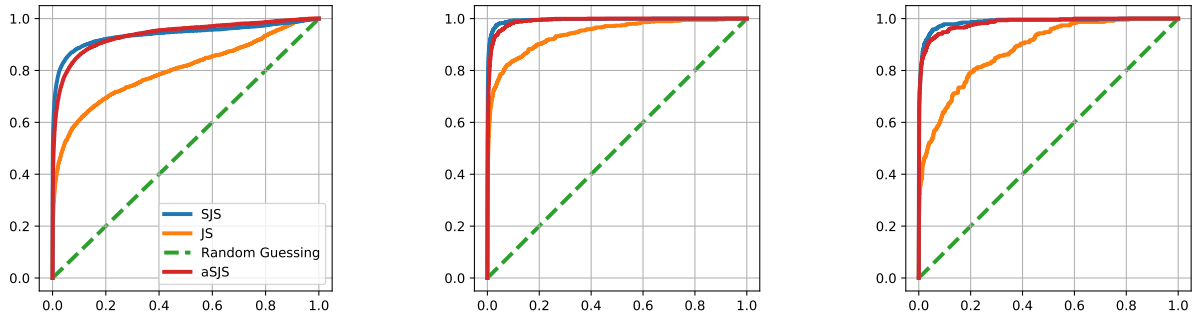


Figure 15: Singular values of empirical centered min-hash matrix $A_i - \mathbf{1}\mathbf{1}^\top$ for PacBio *E. coli* K-12 dataset. By repeating the computation of the singular values of $A_i - \mathbf{1}\mathbf{1}^\top$ for $i = 1, 2, 3, ...$ (i.e., based on different reference reads), we can compute the mean and standard deviation for each sorted singular value, which we display here using error bars of 1σ.



(a) (b) (c)

Figure 16: Analogue of Figure 7 but with minimap2 as ground truth alignments instead of Daligner. We plot ROC curves across different PacBio datasets and different θ thresholds: (a) shows ROC of *E. coli* (K-12 from PacBio website) for alignment threshold $\theta = 0.3$, (b) shows ROC of *E. coli* for alignment threshold $\theta = 0.8$, and (c) shows ROC of *K. Pneumoniae* (NCTC5047) for alignment threshold $\theta = 0.8$



(a) (b) (c)

Figure 17: Analogue of Figure 7 with aSJS added (one step of power iteration approximation to SJS) on Daligner. We plot ROC curves across different PacBio datasets and different θ thresholds: (a) shows ROC of *E. coli* (K-12 from PacBio website) for alignment threshold $\theta = 0.3$, (b) shows ROC of *E. coli* for alignment threshold $\theta = 0.8$, and (c) shows ROC of *K. Pneumoniae* (NCTC5047) for alignment threshold $\theta = 0.8$

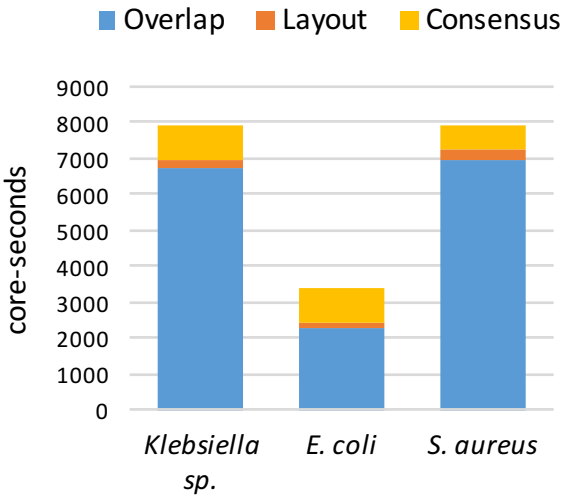


Figure 18: Time required for different stages of the assembly pipeline for different bacterial genomes. Note that the vast majority of time is spent determining pairwise alignments in the overlap stage (Adapted from Figure S11 in Kamath et al. [2017]).