Research Proposal: *Decentralized indexes for genomic data*

Luiz Irber[1]

[1]University of California, Davis

04 December 2018

Introduction

Background

Problem Description

Biological journals require data to be deposited in public sequence databases. Initially they were published in Genbank, but with NGS the volume of data increased and NCBI created the Sequence Read Archive was created to serve as initial stop for archival, with other databases deriving and processing the data more.

Currently , the NCBI Sequence Read Archive (SRA) contain more than 6 Petabases, but tools like MegaBLAST can only search a subset of all the experiments. Queries are also limited in size, since MegaBLAST is an alignment-based algorithm. Question like "What experiments are similar to mine?" are hard to answer in this context,

Scaled MinHashes

The Jaccard similarity of two sets is defined as

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

.

The MinHash [5] sketch was developed at Altavista in the context of document clustering and deduplication. It provides an estimate of the Jaccard similarity (called **resemblance** by [5]) by applying a hash function $h(x)$ to the *w-shingling* of a document D, where a *shingling* is a continuous subsequence of words contained in D, and keeping the n smallest hash values as a representative sketch of the original document. The article also defines the **containment** of two documents as

$$C(A, B) = \frac{|A \cap B|}{|A|}$$

, which measures how much of

Mash [12] was the first implementation of MinHashes in the genomic context. Mash needs extra information (the genome size for the organism being queried) to account for genomic complexity in datasets. This extra information is required because using a fixed size minhash leads to different degrees of accuracy when comparing across highly-diverged organisms (bacteria to animals, for example), and it is even more extreme when taking more complex datasets into account (like metagenomes).

sourmash [18] is another implementation of MinHashes in genomic contexts, adding mainly two variations to the basic method: scaled (or banded) minhashes, and keeping track of hash abundances.

Frequency moments [2]

HyperLogLog [8] HyperLogLog++ [9] KmerStream [10] ntCard [11]

HyperMinHash [21]

HistoSketch [20] HULK [14]

Hierarchical index structures

Linear searching of MinHashes is not practical well when hundreds of thousands of datasets are available. One solution to this problem is to use an hierarchical index structure like Bloofi [6],

Sequence Bloom Tree [15], Split Sequence Bloom Trees [16] AllSome Sequence Bloom Trees [17]

k-mer mapping index structures

Other approaches for indexing include BIGSI [4] and Mantis [13], which avoid the data duplication in SBTs by storing mappings of k-mers to specific datasets directly. In the case of Mantis updating the index leads to more colors (experiments containing the specific k-mer) being tracked, and it is still not clear how that would scale for very large databases. Both approaches still focus on indexing the full k-mer set, but can be adapted to work with MinHash too.

Decentralized querying

IPFS [3], Persistent Data Structures [7], SRA closure [1], A little centralization [19]

Aims

1. **Using scaled MinHash for abundance distribution and cardinality estimation.** The scaled MinHash already allows comparing datasets with distinct genomic complexity, but it is still limited to similarity operations. I propose to extend it to support cardinality estimation using an approach derived from the HyperLogLog cardinality estimator, and also as an approximate abundance distribution estimator. The goal is to have one sketch supporting these three operations, even though other approaches (like the HLL for cardinality estimation or ntCard for abundance distribution) are more appropriate for a specific question.

   Both operation leverage the existing support for tracking abundances for each value in the MinHash, and the scaled approach makes it easier to give better guarantees of the result because we know how 'full' the band is.

2. **Fast queries on many MinHashes using MinHash Bloom Trees** A MinHash Bloom Tree (MHBT) is similar to a Sequence Bloom Tree (SBT), but using a MinHash to represent a dataset instead of Bloom Filters containing the full k-mer spectrum. Leaves are still datasets, with internal nodes representing the union of all hashes present in the MinHashes below it. The preliminary implementation already support searching and insertion of new datasets, but insertion is naive (add to the next available space). I propose to implement insertion based on maximizing shared hashes under an internal node, since this allows faster pruning in the search space when querying.

   While the SBT still adheres to presence filtering like Bloofi (using Bloom Filters for internal nodes) there are other useful data structures that can be used instead and allow a wider range of operations. Multiset representations allow keeping track of the hashes abundances, so using a Count-Min Sketch or a Counting Quotient Filter to represent internal nodes allow other useful queries in the tree. But there is an additional consideration in this case: how to calculate the union of these data structures.

Usually the union of two multisets is defined as the sum of abundances of each multiset for a specific element, but in the hierachical index this leads to not-so-useful queries (the root node would have an abundance count of how many times a hash happened in all datasets). We can use other definitions of the union to create more useful queries:

```
Similarity: with Bloom Filters;
Abundance: using max Count-Min Sketch;
Occurrence: Using "Counting" CMS / CQF
```

Since these new definitions for the union maintain the hashes untouched, this means that an optimal tree structure can be shared among all trees, independent of what kind of internal node is used. This leads to the result that a bare tree (containing only the leaves and the a representation of the tree structure, but not the content of the internal nodes) is enough to build all the other variations of the index. In network-restricted environments (where it is cheaper to rerun the creation of internal nodes data instead of transferring it) this can also lead to more efficient use of resources without loosing generality. Also, if no additional insertions to the index are expected, this can also serve as the backbone for more efficient representations (in a sense this is what the SSBT is to a SBT).

OTHER IDEAS: MHBT <-> RevIndex (dual), establish conversions between indexes. SSBT and AllSome are specializations of MHBT, Mantis and BIGSI are specializations of RevIndex. Support a basic Index API for them, let the user optimize depending on the application.

3. **Decentralized indices for genomic data.** The structure of a MinHash Bloom Tree can be thought of as a persistent data structure: each leaf in the tree never change, and for each insertion $O(logn)$ internal nodes (the internal nodes between the leaf and the root) need to be updated. Since all the other internal nodes (and the leaves) will still be the same as the tree before the update, this view of the MHBT as a persistent data structure makes it a very good fit for storing it in a Merkle Tree.

   I'll explore two different decentralized data storage systems (`IPFS` and `Dat`) as ways of storing and interacting with MHBT indices. - Popular indices benefit from increase bandwidth for downloading data - Derived indices still benefit from nodes shared with the original index

   On top the data storage aspects, another important point is how researchers can interact with these indices (both querying and updating it) in a way that a central authority is not essential (but operations are optimized if it is). This is important in the context of long term sustainability of the system, ~~since I hope to graduate one day and I can't promise I will maintain the system~~ something often overlooked in bioinformatics systems. The initial implementation will be centralized for simplicity and prototyping the user interaction, supporting data submission and querying, with a data pipeline based on the experience downloading 400k+ microbial datasets from NCBI SRA and also the preparation of indices for the IMG database from JGI (60k+ datasets). Once this prototype is functional and we find what features are desirable, I plan to start decentralizing this process by defining updates in terms of CRDT operations, publishing a feed of these changes and moving to PubSub messaging between remote sites, allowing them to update their indices and keep them consistent with ours.

   Because these systems are content-aware, modifying a signature (the JSON file containing the MinHash + metadata) leads to different addresses on the network, which is suboptimal for data sharing. I also plan to explore how to use other systems to link back and provide additional metadata (for example: taxonomy records) using IPLD (Interplanetary Linked Data, a format similar to JSON-LD but focusing on IPFS) and also Hypothesis, a web annotation tool.

References

[1] Closure of the NCBI SRA and implications for the long-term future of genomics data storage. 12:402. ISSN 1474-760X. doi: 10.1186/gb-2011-12-3-402. URL http://dx.doi.org/10.1186/gb-2011-12-3-402.

[2] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 20–29. ACM. URL http://dl.acm.org/citation.cfm?id=237823.

[3] J. Benet. IPFS - content addressed, versioned, p2p file system. URL http://arxiv.org/abs/1407.3561.

[4] P. Bradley, H. d. Bakker, E. Rocha, G. McVean, and Z. Iqbal. Real-time search of all bacterial and viral genomic data. page 234955. doi: 10.1101/234955. URL https://www.biorxiv.org/content/early/2017/12/15/234955.

[5] A. Z. Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings*, pages 21–29. IEEE. URL http://ieeexplore.ieee.org/abstract/document/666900/.

[6] A. Crainiceanu and D. Lemire. Bloofi: Multidimensional bloom filters. 54:311–324. ISSN 03064379. doi: 10.1016/j.is.2015.01.002. URL http://arxiv.org/abs/1501.01941.

[7] J. R. Driscoll, N. Sarnak, D. D. Sleator, and R. E. Tarjan. Making data structures persistent. 38 (1):86–124. ISSN 0022-0000. doi: 10.1016/0022-0000(89)90034-2. URL http://www.sciencedirect.com/science/article/pii/0022000089900342.

[8] P. Flajolet, . Fusy, O. Gandouet, and F. Meunier. HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm. (1). URL http://www.dmtcs.org/dmtcs-ojs/index.php/proceedings/article/viewArticle/914.

[9] S. Heule, M. Nunkesser, and A. Hall. HyperLogLog in practice: Algorithmic engineering of a state of the art cardinality estimation algorithm. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 683–692. ACM.

[10] P. Melsted and B. V. Halldórsson. KmerStream: streaming algorithms for k -mer abundance estimation. 30(24):3541–3547. ISSN 1367-4803. doi: 10.1093/bioinformatics/btu713. URL https://academic.oup.com/bioinformatics/article/30/24/3541/2422237/KmerStream-streaming-algorithms-for-k-mer.

[11] H. Mohamadi, H. Khan, and I. Birol. ntCard: a streaming algorithm for cardinality estimation in genomics data. doi: 10.1093/bioinformatics/btw832. URL https://academic.oup.com/bioinformatics/article/doi/10.1093/bioinformatics/btw832/2832780/ntCard-a-streaming-algorithm-for-cardinality.

[12] 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. 17:132. ISSN 1474-760X. doi: 10.1186/s13059-016-0997-x. URL http://dx.doi.org/10.1186/s13059-016-0997-x.

[13] P. Pandey, F. Almodaresi, M. A. Bender, M. Ferdman, R. Johnson, and R. Patro. Mantis: A fast, small, and exact large-scale sequence search index. page 217372. doi: 10.1101/217372. URL https://www.biorxiv.org/content/early/2017/11/10/217372.

[14] W. P. Rowe, A. P. Carrieri, C. Alcon-Giner, S. Caim, A. Shaw, K. Sim, J. S. Kroll, L. Hall, E. O. Pyzer-Knapp, and M. D. Winn. Streaming histogram sketching for rapid microbiome analytics. page 408070. doi: 10.1101/408070. URL https://www.biorxiv.org/content/early/2018/09/04/408070.

[15] B. Solomon and C. Kingsford. Fast search of thousands of short-read sequencing experiments. 34(3): 300–302, . ISSN 1087-0156. doi: 10.1038/nbt.3442. URL https://www.nature.com/nbt/journal/v34/n3/full/nbt.3442.html.

[16] B. Solomon and C. Kingsford. Improved search of large transcriptomic sequencing databases using split sequence bloom trees. In *International Conference on Research in Computational Molecular Biology*, pages 257–271. Springer, .

[17] C. Sun, R. S. Harris, R. Chikhi, and P. Medvedev. AllSome sequence bloom trees. In *Research in Computational Molecular Biology*, Lecture Notes in Computer Science, pages 272–286. Springer, Cham. ISBN 978-3-319-56969-7 978-3-319-56970-3. doi: 10.1007/978-3-319-56970-3_17. URL https://link.springer.com/chapter/10.1007/978-3-319-56970-3_17.

[18] C. Titus Brown and L. Irber. sourmash: a library for MinHash sketching of DNA. 1(5). doi: 10.21105/joss.00027. URL http://joss.theoj.org/papers/10.21105/joss.00027.

[19] J. N. Tsitsiklis and K. Xu. On the power of (even a little) centralization in distributed processing. 39 (1):121–132.

[20] D. Yang, B. Li, L. Rettig, and P. Cudre-Mauroux. HistoSketch: Fast similarity-preserving sketching of streaming histograms with concept drift. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 545–554. ISBN 978-1-5386-3835-4. doi: 10.1109/ICDM.2017.64. URL doi.ieeecomputersociety.org/10.1109/ICDM.2017.64.

[21] Y. W. Yu and G. Weber. HyperMinHash: Jaccard index sketching in LogLog space. URL http://arxiv.org/abs/1710.08436.