Received Date: June 2023 Revised: Aust 2023 Accepted: October 2023

Performance Analysis of Sketching Methods

- Dinesh Maharjan[1]

Abstract

One of the problems in data mining or machine learning is the high-dimensional dataset. MinHash can generate sketches of sparse datasets efficiently reducing the dimension to a few thousand. These sketches, then, can be used for different machine-learning applications. It takes O(kd) computations to generate k hash values for a data point with d non-zeros. The Sketch of a data point is the vector of k hash values. Weighted Minwise Hashing is another method to generate a sketch of size k in O(kp/d) computations. Here, p is the size of the universal set. Optimal Densification is the most efficient and accurate method as it can generate k hash values in mere O(d + k) computations. In this paper, we investigate the performance of Optimal Densification, Weighted Minwise Hashing, and Vanilla MinHash by performing two experiments. Firstly, we investigate Jaccard similarity estimation accuracy on six different synthetic datasets. Then, we perform one nearest neighbor classification (1NN) of four real datasets. Optimal Densification outperforms both Weighted Minwise Hashing and Vanilla MinHash in terms of accuracy and time taken to generate the sketches.

Keywords: MinHash, Sketch, Jaccard Similarity, Densification

Introduction

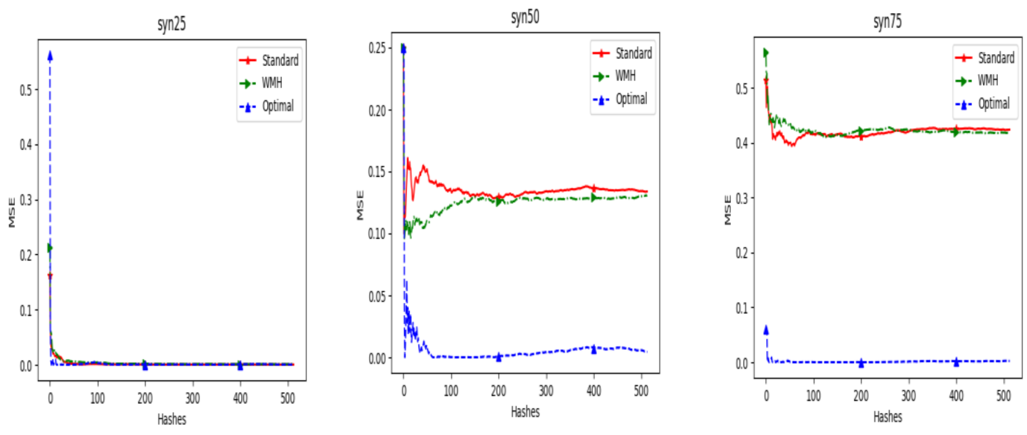
Similarity computation is the basic operation that needs to be performed in machine learning, data mining, image recognition, and speech analysis for the comparison of data points. One of the comparison metrics is Jaccard similarity. For the universal set

1 Mr. Maharjan is a Ph.D. Scholar at the University of Seoul, Seoul, South Korea.

$U = \{0, 1, 2, \ldots, p - 1\}$, the Jaccard similarity between two sets *S* and *T* is defined in Eq. 1.

$J = \frac{S \cap T}{S \cup T}$……………………….. 1

It is the fraction of similar items between two sets. So Jaccard similarity can be used in many applications. However, computing Jaccard similarity for datasets with large *p* is very expensive. In the worst case, it takes *O(p)* time. So Broder introduced MinHash (Broder, Charikar, Frieze, & Mitzenmacher, 2000) which reduces the dimension from *p* to *k* by generating high-quality sketches. Here $p >> k$. Sketches of two sets *S* and *T* can be used to estimate Jaccard similarity. The most important applications of sketches generated by MinHash are indexing streams (Mahadevan, 2000), classifying text (Chi, Li, & Zhu, 2014), clustering images (C. Li, Chang, Garcia-Molina, & Wiederhold, 2002), and Locality Sensitive Hashing (Andoni & Indyk, 2006; Andoni, Indyk, Laarhoven, Razenshteyn, & Schmidt, 2015; Har-Peled, Indyk, & Motwani, 2012). So in recent years, many semantic works have been published related to MinHash. Originally MinHash (Broder, 1997; Broder et al., 2000) is designed to quickly detect the near-identical text documents on the basis of Jaccard similarity. However, it can be used for estimating cosine similarity (Shrivastava and Li, 2014c) also. Thus, MinHash is very popular in large-scale data mining techniques (Chum et al., 2010; Tamersoy, Roundy, & Chau, 2014). Although MinHash achieves computational improvement over the naive approach, it is not sufficient for high-dimensional datasets. This is because, given a universal set *U* of size *p*, it needs to generate the permutation of size *p* to obtain each hash value. Moreover, it requires *O(kd)* computations to generate a sketch of size *k,* where *d* is the number of non-zeros in *S*. Thus, generating a hash value becomes the major bottleneck for large *k* and *d*. One Permutation Hashing (OPH) (P. Li, Owen, & Zhang, 2012) is able to generate the same number of hash values with a single permutation and gains remarkable improvement in execution time. It needs only *O(k + d)* computations. However, it is unable to generate hash values for empty bins that do not contain any non-zero elements. This issue is addressed in papers (Shrivastava & Li, 2014a, 2014b) with the method of densification. Their method lacks enough randomness, and as a result, the estimated Jaccard Similarity has high variance. Optimal densification (Shrivastava, 2017) successfully solves the empty bin problem with sufficient randomness by using a 2-universal hashing function.



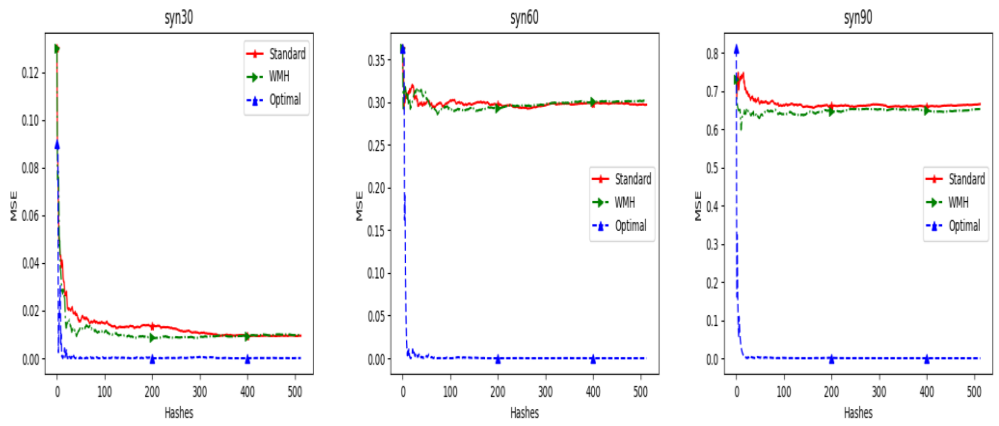


Figure 1: MSE with the size of sketch, k. The MSE values are averaged over 10 executions.

We compare the performance of MinHash, Weighted Minwise Hashing, and Optimal Densification by performing two tasks on synthetic as well as real datasets. The real datasets are downloaded from (Hsu et al., 2003) and most of these datasets are never used in the literature.

Objectives of the Study

The main purpose of this study is to investigate the performance of different state-of-art methods for reducing dimension. The special objective of the study follows below:

1. To examine the performance of MinHash, Weighted Min-wise Hashing, and Optimal Densification on time series data.
2. To use the sketches generated by hash-based methods as the features for the classification of time series data.

Review of Sketching Methods

We review Standard MinHash, Weighted MinHash, and Optimal Densification. MinHash paved the foundation for hash-based dimensionality reducing methods.

Weighted Minwise Hasing and Optimal Densification are efficient and effective versions of MinHash,

Standard MinHash

Standard MinHash (Broder, 1997; Broder et al., 2000) is an LSH (Locality Sensitive Hashing) randomized hashing method that is widely used for big data analysis. It improves the naive method of computing Jaccard similarity by generating small-sized sketches of high-dimensional sets. For the universal set $U = \{0, 1, . . . , p - 1\}$, it generates a random permutation, π, of size p and applies it to a set S. Then, the minimum among the permutation values of the elements in S is returned as a hash value. That is,

$$h_\pi^{standard}(S) = min(\pi(S))$$

A new independent permutation is generated to obtain each hash value. MinHash, as a consistent hashing scheme, uses the same permutation to generate the jth hash value of all the sets. The probability of collision is equal to the Jaccard similarity itself i.e. $P(h_\pi^{standard}(S)=h_\pi^{standard}(T)) = J(S, T)$, where $h_\pi^{standard}(S)$ and $h_\pi^{standard}(T)$ are hash values generated by executing MinHash for sets S and T respectively. MinHash must perform $O(kd)$ computations to generate sketches of size k. The value of k and d can explode up to thousands and millions respectively. In such cases, MinHash performs very slowly.

Optimal Densification of OPH

Shrivastava and Li introduced the method of densification (Shrivastava & Li, 2014b) to improve Standard MinHash by randomly sampling the direction of densification. The direction may be circularly right or circularly left depending upon the discrete random number generated in [0, 1]. Such densification lacks sufficient randomness, and thus, has high variance on highly sparse datasets. For fast and accurate sketching, Optimal densification (Shrivastava, 2017) generates a random number, z_i for each $i \in S$ using a 2-universal hash function (Dietzfelbinger, Hagerup, Katajainen, & Penttonen, 1997) as shown in Eq. 2.

$$z_i = (a_i + b) >> l - w \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 2$$

, where a and b are random odd integers generated from a uniform distribution, and l and w are bit sizes of i and z_i respectively. It is worth noting that Eq. 2 helps to gain sufficient randomness by randomly converting a l-bit i into a w-bit z_i. z_i works as the permutation value for the corresponding i. Optimal densification, then, splits $[0, 2w)$ into k bins of equal size. After generating z_i for all i in a bin, it returns a minimum z_i as the hash value for the bin. Empty bins, j, are assigned with the hash value of non-empty bins using Eq. 3.

$$j = (a_j + b) >> l - k \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 3$$

, where $a_j + b$ is of l-bits. Instead of sampling non-empty bins from a predetermined direction as done in (Shrivastava & Li, 2014b), Eq. 3 uniformly samples non-empty bins from a random direction. This provides sufficient randomness and makes optimal densification efficient and accurate. Since its time complexity to generate k hash values is $O(k + d)$, it will be slow for large d. Weighted Minwise Hashing Weighted Minwise Hashing (WMH) (Shrivastava, 2016) is a generalization of Standard MinHash and achieves computational improvements over MinHash on sets with $d \geq 0.5p$. It can generate sketches of binary sets as well as weighted sets. Given a universal set, $U = \{0, 1, \ldots, p - 1\}$, WMH estimates Jaccard similarity between two sets S and

T by generating a sketch of each set using Rejection Sampling (Casella, Robert, Wells, et al., 2004). In order to generate hash values consistently, WMH assigns a unit weight to each non-zero element of *S*. It means upper bounds, b_i, for each element i in the universal set *U* is 1. Then, cumulative upper bounds, $r_i \in \{0, 1, 2, . . . , p\}$. Notice that, the corresponding non-zero elements of each set fall into the same interval $[r_i, r_i+1)$. If i ∈ S, then $[r_i, r_i+1)$ is considered as the white map otherwise black map. WMH attempts to sample the white map by generating Continuous Uniform Random variable x in $[0, p)$. If *x* lies in the white map, the number of attempts, *c,* made to sample the white map is returned as a hash value. If *x* lies in the black map, then it resamples by generating another Uniform random number. This process goes on until the white map is sampled. WMH also satisfies the LSH property i.e.

$$p(h^{wmh}(S) = h^{wmh}(T)) = J(S, T)$$

Note that $h^{wmh}(S)$ is the sketch generated by executing the WMH algorithm for *S*.

WMH is summarized in Algorithm 1.

Algorithm 1 Weighted Minwise Hashing

Input: set, S, size of sketch, k, random seeds, seed, size of universal set, p

Output: a sketch of S, h

```
1: for j = 0, 1, . . . , k − 1 do
2: c=0
3: set $seed_j$
4: while true do
5: x ∼ Continuous Uniform (0, 1)
6: i = $\lfloor x * p \rfloor$
```

```
7: if i ∈ S then
8: break
9: end if
10: c=c+1
11: end while
12: $h_j = c$
13: end for
14: return h
```

Rejection sampling keeps on trying to sample a white map until it achieves the first success. Thus, the hash value, $h^{wmh}(S)$ has geometric distribution with probability of success d/p. The expected value of $h^{wmh}(S)$ is p/d. Thus, the time complexity of WMH to generate k hash values is O(pk/d).

Table 1: Basic Statistics of Datasets

Dataset	Train Set Size	Test Set Size	P	Sparsity
Syn30	1	1	1024	0.6
Syn60	1	1	1024	0.6
Syn90	1	1	1024	0.6
Syn25	1	1	1024	0.8
Syn50	1	1	1024	0.8
Syn75	1	1	1024	0.8
StarLightCurves	8236	1000	1024	0.54
HandOutlines	1000	370	2709	0.41
Gisette	6000	1000	5000	0.87
Forde	3601	1320	500	0.5

For a sparse dataset, the value of *d* will be very small compared to *p* which leads to a large expected value of $h^{wmh}(S)$. Thus, WMH becomes very slow on highly sparse datasets. However, if $d >= 0.5p$ and $d \leq k$, then $O(pk/d) < O(k + d)$. In such cases, WMH will execute faster than Optimal densification.

Locality Sensitive Hashing (LSH):

Locality Sensitive Hashing (LSH) (Andoni and Indyk, 2006) is one of the applications of mapping sets into sketches. It hashes similar sets to the same bucket with high probability which reduces search space to a great extent. It means sets with the same elements or high similarity will be hashed with the same code with high probability. We then compute the similarity only among the sets which got hashed into the same bucket. As we have already mentioned sketch is the vector of hash codes extracted for a given set using different hash functions, MinHash (Broder, 1997; Broder et al., 2000) was proposed as the LSH method of Jaccard similarity for the quick detection of similar text documents. A. Shrivastava claims that MinHash can be used to estimate the Cosine similarity also (Charikar, 2002).

Weighted Set:

The major drawback of MinHash is that it can deal with binary sets only not with the weighted sets. A weighted set is very common in real-life problems as it can represent the importance of an element in the set. A weighted set S contains weights s_i for the elements i. Given a universal set, $\Omega = \{0, 1, . . . , p - 1\}$, Generalized Jaccard Similarity (GJS) between two weighted sets $S = \{s_0, s_1, s_2, . . . , s_{p-1} \}$ and $T = \{t_0, t_1, t_2, . . . , t_{p-1} \}$ introduced in (HAVELIWALA, 2000) and was efficiently estimated by introducing the method of unweighted minwise hashing. Their method (HAVELIWALA, 2000) converts weighted set into unweighted by expanding the s_i with additional distinct elements $\{s_1, s_2, . . . , s_{s_i} \}$ in proportion to its weight s_i. Then Standard MinHash

(Broder et al., 2000) is applied. This approach cannot deal with real weights and is inefficient. Other unweighted minwise hashing methods (Gollapudi and Panigrahy, 2006; Haeupler et al., 2014) can deal with the real weights but they still have to generate additional elements. A significant improvement was seen in the paper (Manasse et al., 2010) that uses pairs of active indices (w_i, z_i) to generate each hash value. Their method is referred to as Consistent Weighted Sampling (CWS). However, this approach still needs to sample many active indices per weight to obtain each hash value.

Research Methods

We use theoretical and empirical research methods. We first theoretical correctness of Vanilla MinHash, Weighted Min-wise Hashing, and Optimal Densification in estimating Jaccard similarity. Then, we perform experiments to prove the unbiasedness of all the methods. The experiments use six synthetic datasets and three-time series datasets. The experimental results prove that all of the methods are unbiased estimators of Jaccard similarity. Moreover, sketches can be used for machine learning purposes.

Experimental Results

In order to perform a comparative study between Bitwise MinHash and other state-of-the-art methods, Standard MinHash (Broder et al., 2000), Optimal Densification (Shrivastava, 2017), and WMH (Shrivastava, 2016), we perform two different tasks on synthetic and real datasets. We first inspect consistency in estimating Jaccard similarity using six pairs of synthetic data points. The pairs are of different similarity and sparsity. Then, we investigate the efficiency and effectiveness of compared methods with 1-NN classification. The Basic statistics on the datasets used for our experiments is provided in Table 1.

MSE in estimating Jaccard Similarity

Consistency is one of the important qualities of an estimator. Thus, we investigate the consistency of compared methods on six synthetic datasets. We record the Mean Squared Error (MSE) of compared methods for each hash value. MSE measures the difference between actual Jaccard similarity and estimated Jaccard similarity. Moreover, we compare the efficiency of each method on the basis of hash generation time. Results are shown in Fig. 1.

Synthetic datasets, Syn30, Syn60, Syn90, Syn25, Syn50, and Syn75 are pairs of data points created by sampling elements from a uniform distribution at intervals [0, 1024). The Jaccard similarity between the pair of data points in Syn30 is 0.30 and the same applies to other synthetic datasets.

Discussion on the Results: MSE of all methods in Fig. 1 decreases with an increase in sketch size, *k*. This ensures the consistency of all the estimators. This also proves that bias is negligible. Optimal Densification beats all other methods in terms of MSE. The time taken for generating sketches is shown in Fig. 2. Results show that Optimal Densification is almost 20 times faster than Standard MinHash.

1-NN Classification

We conduct 1-NN classification on four real datasets, StarLightCurves, Han Outlines, Gisette, and FordA to further assess the quality of signatures. All datasets are downloaded from (Chen et al., 2015). We compare the effectiveness of all the methods on the basis of classification accuracy. Accuracy is the proportion of correctly classified test sets. The plots for the classification accuracy are shown in Fig. 2.

Discussion on the Results: We can see that Optimal Densification beats both WMH and Standard MinHash. WMH and Standard MinHash stand at similar classification accuracy in three datasets. However, WMH has better accuracy than Standard MinHash in case of StarLightCurves.

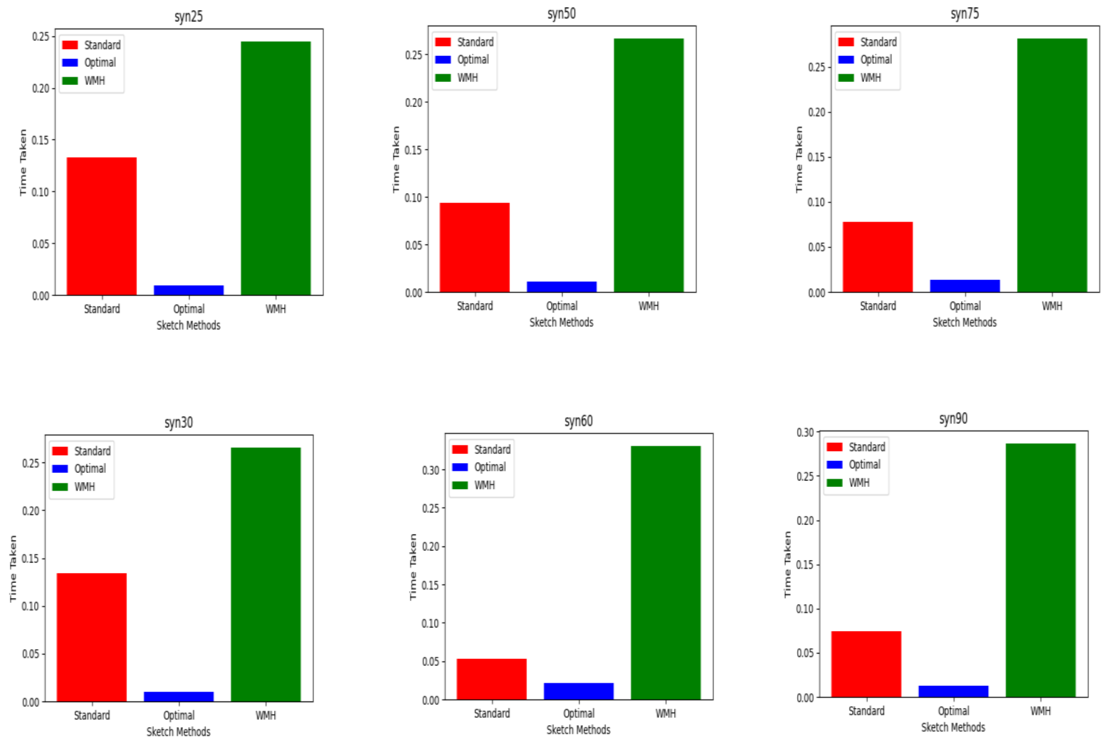


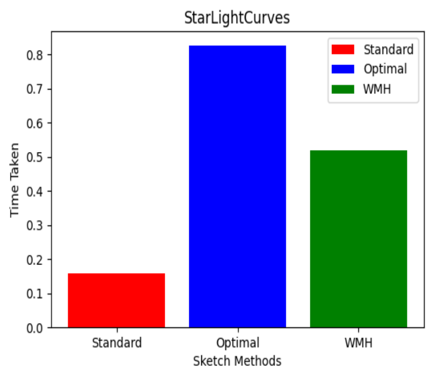
Figure 2. Time taken to generate hash with the size of a sketch, *k*. The Time taken is averaged over 10 executions.

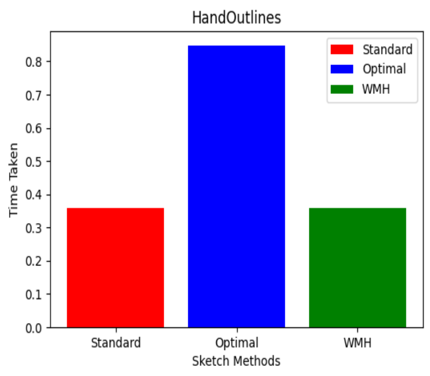
Conclusion

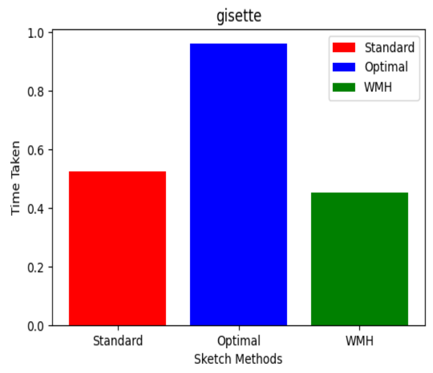
In this paper, we compare the quality of sketches generated by Standard MinHash, Optimal Densification, and WMH. We calculate estimated Jaccard similarity using the sketches and compare the Mean Squared Errors. Results show that Optimal

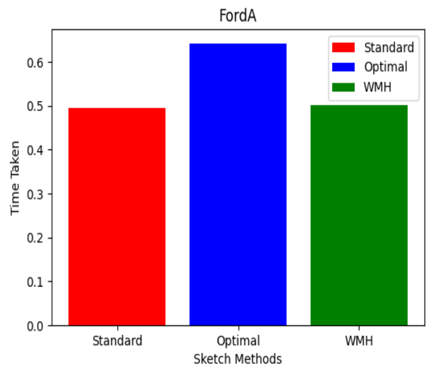
Densification outperforms both Standard MinHash and WMH. Moreover, Optimal Densification is the fastest among the three in generating sketches.

The best application of sketch is to classify the data point. We perform the One Nearest Neighbor (1NN) classification of four real datasets by computing the Jaccard similarity of each test point with the training set. The test data point is classified with the class of the train data point having maximum similarity. As expected, Optimal Densification has better classification accuracy. Thus, we conclude that Optimal Densification is one of the most accurate and efficient similarity search methods.









References

Andoni, A., & Indyk, P. (2006). Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In Foundations of computer science, 2006. focs'06. 47th annual ieee symposium on (pp. 459–468). IEEE.

Andoni, A., Indyk, P., Laarhoven, T., Razenshteyn, I., & Schmidt, L. (2015). Practical and optimal lsh for angular distance. In Advances in neural information processing systems (pp. 1225–1233).

Broder, A. Z. (1997). On the resemblance and containment of documents. In Compression and complexity of sequences 1997. proceedings (pp. 21–29). IEEE.

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

Casella, G., Robert, C. P., Wells, M. T., et al. (2004). Generalized accept-reject sampling schemes. In A festschrift for herman rubin (pp. 342–347). Institute of Mathematical Statistics.

Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A., & Batista, G. (2015). The ucr time series classification archive. www.cs.ucr.edu /~eamonn/time_series_data/

Chi, L., Li, B., & Zhu, X. (2014). Context-preserving hashing for fast text classification. In Proceedings of the 2014 siam international conference on data mining (pp. 100–108). SIAM.

Chum, O. et al. (2010). Large-scale discovery of spatially related images. IEEE transactions on pattern analysis and machine intelligence, 32(2), 371–377.

Dietzfelbinger, M., Hagerup, T., Katajainen, J., & Penttonen, M. (1997). A reliable randomized algorithm for the closest-pair problem. Journal of Algorithms, 25(1), 19–51.

Har-Peled, S., Indyk, P., & Motwani, R. (2012). Approximate nearest neighbor: Towards removing the curse of dimensionality. Theory of computing, 8(1), 321–350.

Li, C., Chang, E., Garcia-Molina, H., & Wiederhold, G. (2002). Clustering for approximate similarity search in high-dimensional spaces. IEEE Transactions on Knowledge and Data Engineering, 14(4), 792–808.

Li, P., Owen, A., & Zhang, C.-H. (2012). One permutation hashing. In Advances in neural information processing systems (pp. 3113–3121).

Mahadevan, B. (2000). Business models for internet-based e-commerce: An anatomy. California management review, 42(4), 55–69.

Shrivastava, A. (2016). Simple and efficient weighted minwise hashing. In Advances in neural information processing systems (pp. 1498–1506).

Shrivastava, A. (2017). Optimal densification for fast and accurate minwise hashing. arXiv preprint arXiv:1703.04664.

Shrivastava, A., & Li, P. (2014a). Densifying one permutation hashing via rotation for fast near neighbor search. In International conference on machine learning (pp. 557–565).

Hsu, C.W., Chang, C.C., Lin, C.J., et al., 2003. A practical guide to support vector classification. Technical Report. URL: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.

Shrivastava, A., & Li, P. (2014b). Improved densification of one permutation hashing. arXiv preprint arXiv:1406.4784.

Shrivastava, A., & Li, P. (2014c). In defense of minhash over simhash. In Artificial intelligence and statistics (pp. 886–894).

Tamersoy, A., Roundy, K., & Chau, D. H. (2014). Guilt by association: Large scale malware detection by mining file-relation graphs. In Proceedings of the 20th acm sigkdd international conference on knowledge discovery and data mining (pp. 1524–1533). ACM.