



WIKIPEDIA  
The Free Encyclopedia

[Main page](#)  
[Contents](#)  
[Featured content](#)  
[Current events](#)  
[Random article](#)  
[Donate to Wikipedia](#)  
[Wikipedia store](#)

Interaction

[Help](#)  
[About Wikipedia](#)  
[Community portal](#)  
[Recent changes](#)  
[Contact page](#)

Tools

[What links here](#)  
[Related changes](#)  
[Upload file](#)  
[Special pages](#)  
[Permanent link](#)  
[Page information](#)  
[Wikidata item](#)  
[Cite this page](#)

Print/export

[Create a book](#)  
[Download as PDF](#)  
[Printable version](#)

Languages


[Español](#)  
[فارسی](#)  
[中文](#)

 [Edit links](#)

[Create account](#) [Log in](#)

Article [Talk](#)

[Read](#) [Edit](#) [View history](#)



# MinHash

From Wikipedia, the free encyclopedia

In [computer science](#), **MinHash** (or the **min-wise independent permutations locality sensitive hashing** scheme) is a technique for quickly estimating how [similar](#) two sets are. The scheme was invented by [Andrei Broder](#) (1997),<sup>[1]</sup> and initially used in the [AltaVista](#) search engine to detect duplicate web pages and eliminate them from search results.<sup>[2]</sup> It has also been applied in large-scale [clustering](#) problems, such as [clustering documents](#) by the similarity of their sets of words.<sup>[1]</sup>

## Contents [hide]

- [Jaccard similarity and minimum hash values](#)
- [Algorithm](#)
  - [2.1 Variant with many hash functions](#)
  - [2.2 Variant with a single hash function](#)
  - [2.3 Time analysis](#)
- [Min-wise independent permutations](#)
- [Applications](#)
- [Other uses](#)
- [Evaluation and benchmarks](#)
- [See also](#)
- [External links](#)
- [References](#)

## Jaccard similarity and minimum hash values [\[edit\]](#)

The [Jaccard similarity coefficient](#) is a commonly used indicator of the similarity between two sets. For sets *A* and *B* it is defined to be the ratio of the number of elements of their [intersection](#) and the number of elements of their union:

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

This value is 0 when the two sets are [disjoint](#), 1 when they are equal, and strictly between 0 and 1 otherwise.

Two sets are more similar (i.e. have relatively more members in common) when their Jaccard index is closer to 1. It is our goal to estimate *J*(*A*,*B*) quickly, without explicitly computing the intersection and union.

Let *h* be a [hash function](#) that maps the members of *A* and *B* to distinct integers, and for any set *S* define *h*<sub>min</sub>(*S*) to be the minimal member of *S* with respect to *h*—that is, the member *x* of *S* with the minimum value of *h*(*x*). Now, if we apply *h*<sub>min</sub> to both *A* and *B*, we will get the same value exactly when the element of the union *A* ∪ *B* with minimum hash value lies in the intersection *A* ∩ *B*. The probability of this being true is the ratio above, and therefore:

$$\Pr[ h_{\min}(A) = h_{\min}(B) ] = J(A,B),$$

That is, the [probability](#) that *h*<sub>min</sub>(*A*) = *h*<sub>min</sub>(*B*) is true is equal to the similarity *J*(*A*,*B*), assuming randomly chosen sets *A* and *B*. In other words, if *r* is the [random variable](#) that is one when *h*<sub>min</sub>(*A*) = *h*<sub>min</sub>(*B*) and zero otherwise, then *r* is an [unbiased estimator](#) of *J*(*A*,*B*). *r* has too high a [variance](#) to be a useful estimator for the Jaccard similarity on its own—it is always zero or one. The idea of the MinHash scheme is to reduce this variance by averaging together several variables constructed in the same way.

## Algorithm [\[edit\]](#)

### Variant with many hash functions [\[edit\]](#)

The simplest version of the minhash scheme uses *k* different hash functions, where *k* is a fixed integer parameter, and represents each set *S* by the *k* values of *h*<sub>min</sub>(*S*) for these *k* functions.

To estimate *J*(*A*,*B*) using this version of the scheme, let *y* be the number of hash functions for which *h*<sub>min</sub>(*A*) = *h*<sub>min</sub>(*B*), and use *y*/*k* as the estimate. This estimate is the average of *k* different 0-1 random

variables, each of which is one when  $h_{\min}(A) = h_{\min}(B)$  and zero otherwise, and each of which is an unbiased estimator of  $J(A,B)$ . Therefore, their average is also an unbiased estimator, and by standard [Chernoff bounds](#) for sums of 0-1 random variables, its expected error is  $O(1/\sqrt{k})$ .<sup>[3]</sup>

Therefore, for any constant  $\varepsilon > 0$  there is a constant  $k = O(1/\varepsilon^2)$  such that the expected error of the estimate is at most  $\varepsilon$ . For example, 400 hashes would be required to estimate  $J(A,B)$  with an expected error less than or equal to .05.

### Variant with a single hash function [\[edit\]](#)

It may be computationally expensive to compute multiple hash functions, but a related version of MinHash scheme avoids this penalty by using only a single hash function and uses it to select multiple values from each set rather than selecting only a single minimum value per hash function. Let  $h$  be a hash function, and let  $k$  be a fixed integer. If  $S$  is any set of  $k$  or more values in the domain of  $h$ , define  $h_{(k)}(S)$  to be the subset of the  $k$  members of  $S$  that have the smallest values of  $h$ . This subset  $h_{(k)}(S)$  is used as a *signature* for the set  $S$ , and the similarity of any two sets is estimated by comparing their signatures.

Specifically, let  $A$  and  $B$  be any two sets. Then  $X = h_{(k)}(h_{(k)}(A) \cup h_{(k)}(B)) = h_{(k)}(A \cup B)$  is a set of  $k$  elements of  $A \cup B$ , and if  $h$  is a random function then any subset of  $k$  elements is equally likely to be chosen; that is,  $X$  is a [simple random sample](#) of  $A \cup B$ . The subset  $Y = X \cap h_{(k)}(A) \cap h_{(k)}(B)$  is the set of members of  $X$  that belong to the intersection  $A \cap B$ . Therefore,  $|Y|/k$  is an unbiased estimator of  $J(A,B)$ . The difference between this estimator and the estimator produced by multiple hash functions is that  $X$  always has exactly  $k$  members, whereas the multiple hash functions may lead to a smaller number of sampled elements due to the possibility that two different hash functions may have the same minima. However, when  $k$  is small relative to the sizes of the sets, this difference is negligible.

By standard [Chernoff bounds](#) for sampling without replacement, this estimator has expected error  $O(1/\sqrt{k})$ , matching the performance of the multiple-hash-function scheme.

### Time analysis [\[edit\]](#)

The estimator  $|Y|/k$  can be computed in time  $O(k)$  from the two signatures of the given sets, in either variant of the scheme. Therefore, when  $\varepsilon$  and  $k$  are constants, the time to compute the estimated similarity from the signatures is also constant. The signature of each set can be computed in [linear time](#) on the size of the set, so when many pairwise similarities need to be estimated this method can lead to a substantial savings in running time compared to doing a full comparison of the members of each set. Specifically, for set size  $n$  the many hash variant takes  $O(n k)$  time. The single hash variant is generally faster, requiring  $O(n \log k)$  time to maintain the sorted list of minima.<sup>[citation needed]</sup>

## Min-wise independent permutations [\[edit\]](#)

In order to implement the MinHash scheme as described above, one needs the hash function  $h$  to define a random [permutation](#) on  $n$  elements, where  $n$  is the total number of distinct elements in the union of all of the sets to be compared. But because there are  $n!$  different permutations, it would require  $\Omega(n \log n)$  bits just to specify a truly random permutation, an infeasibly large number for even moderate values of  $n$ . Because of this fact, by analogy to the theory of [universal hashing](#), there has been significant work on finding a family of permutations that is "min-wise independent", meaning that for any subset of the domain, any element is equally likely to be the minimum. It has been established that a min-wise independent family of permutations must include at least

$$lcm(1, 2, \dots, n) \geq e^{n-o(n)}$$

different permutations, and therefore that it needs  $\Omega(n)$  bits to specify a single permutation, still infeasibly large.<sup>[2]</sup>

Because of this impracticality, two variant notions of min-wise independence have been introduced: restricted min-wise independent permutations families, and approximate min-wise independent families. Restricted min-wise independence is the min-wise independence property restricted to certain sets of cardinality at most  $k$ .<sup>[4]</sup> Approximate min-wise independence has at most a fixed probability  $\varepsilon$  of varying from full independence.<sup>[5]</sup>

## Applications [\[edit\]](#)

The original applications for MinHash involved clustering and eliminating near-duplicates among web documents, represented as sets of the words occurring in those documents.<sup>[1][2]</sup> Similar techniques have also been used for clustering and near-duplicate elimination for other types of data, such as images: in the case of

image data, an image can be represented as a set of smaller subimages cropped from it, or as sets of more complex image feature descriptions.<sup>[6]</sup>

In [data mining](#), [Cohen et al. \(2001\)](#) use MinHash as a tool for [association rule learning](#). Given a database in which each entry has multiple attributes (viewed as a 0-1 matrix with a row per database entry and a column per attribute) they use MinHash-based approximations to the Jaccard index to identify candidate pairs of attributes that frequently co-occur, and then compute the exact value of the index for only those pairs to determine the ones whose frequencies of co-occurrence are below a given strict threshold.<sup>[7]</sup>

## Other uses <sup>[edit]</sup>

The MinHash scheme may be seen as an instance of [locality sensitive hashing](#), a collection of techniques for using hash functions to map large sets of objects down to smaller hash values in such a way that, when two objects have a small distance from each other, their hash values are likely to be the same. In this instance, the signature of a set may be seen as its hash value. Other locality sensitive hashing techniques exist for [Hamming distance](#) between sets and [cosine distance](#) between [vectors](#); locality sensitive hashing has important applications in [nearest neighbor search](#) algorithms.<sup>[8]</sup> For large distributed systems, and in particular [MapReduce](#), there exist modified versions of MinHash to help compute similarities with no dependence on the point dimension.<sup>[9]</sup>

## Evaluation and benchmarks <sup>[edit]</sup>

A large scale evaluation has been conducted by [Google](#) in 2006 <sup>[10]</sup> to compare the performance of Minhash and [Simhash](#)<sup>[11]</sup> algorithms. In 2007 Google reported using Simhash for duplicate detection for web crawling<sup>[12]</sup> and using Minhash and [LSH](#) for [Google News](#) personalization.<sup>[13]</sup>

## See also <sup>[edit]</sup>


- [Approximate string matching](#)
- [Rolling hash](#)
- [w-shingling](#)
- [Tabulation hashing](#)
- [Bloom filter](#)
- [Count-Min sketch](#)
- [Set cover problem](#)
- [Levenshtein distance](#)
- [String metric](#)
- [Semantic hashing](#)<sup>[14]</sup>
- [Spectral hashing](#)<sup>[15]</sup>
- [Spherical Hashing](#)<sup>[16]</sup>
- [Winner-Take-all hashing](#)<sup>[17]</sup>










## External links <sup>[edit]</sup>

- [Mining of Massive Datasets, Ch. 3. Finding similar Items](#)
- [Simple Simhashing](#)
- [Set Similarity & MinHash - C# implementation](#)
- [Minhash with LSH for all-pair search \(C# implementation\)](#)
- [MinHash – Java implementation](#)
- [MinHash – Scala implementation and a duplicate detection tool](#)
- [All pairs similarity search \(Google Research\)](#)
- [Distance and Similarity Measures\(Wolfram Alpha\)](#)
- [Nilsimsa hash \(Python implementation\)](#)
- [Simhash](#)

## References <sup>[edit]</sup>

- ↑ <sup>a b c</sup> Broder, Andrei Z (1997), "On the resemblance and containment of documents", *Compression and Complexity of Sequences: Proceedings, Positano, Amalfitan Coast, Salerno, Italy, June 11-13, 1997*  (PDF), *IEEE*, pp. 21–29, doi:10.1109/SEQUEN.1997.666900.
- ↑ <sup>a b c</sup> Broder, Andrei Z.; Charikar, Moses; Frieze, Alan M.; Mitzenmacher, Michael (1998), "Min-wise independent permutations", *Proc. 30th ACM Symposium on Theory of Computing (STOC '98)*, New York, NY, USA: [Association](#)

for *Computing Machinery*, pp. 327–336, doi:10.1145/276698.276781 .

3. <sup>^</sup> Vassilvitskii, Sergey (2011), *COMS 6998-12: Dealing with Massive Data (lecture notes, Columbia university)*  (PDF).
4. <sup>^</sup> Matoušek, Jiří; Stojaković, Miloš (2003), "On restricted min-wise independence of permutations", *Random Structures and Algorithms* **23** (4): 397–408, doi:10.1002/rsa.10101 .
5. <sup>^</sup> Saks, M.; Srinivasan, A.; Zhou, S.; Zuckerman, D. (2000), "Low discrepancy sets yield approximate min-wise independent permutation families", *Information Processing Letters* **73** (1–2): 29–32, doi:10.1016/S0020-0190(99)00163-5 .
6. <sup>^</sup> Chum, Ondřej; Philbin, James; Isard, Michael; Zisserman, Andrew (2007), "Scalable near identical image and shot detection", *Proceedings of the 6th ACM International Conference on Image and Video Retrieval (CIVR'07)*, doi:10.1145/1282280.1282359 ; Chum, Ondřej; Philbin, James; Zisserman, Andrew (2008), "Near duplicate image detection: min-hash and tf-idf weighting", *Proceedings of the British Machine Vision Conference*  (PDF) **3**, p. 4.
7. <sup>^</sup> Cohen, E.; Datar, M.; Fujiwara, S.; Gionis, A.; Indyk, P.; Motwani, R.; Ullman, J. D.; Yang, C. (2001), "Finding interesting associations without support pruning", *IEEE Transactions on Knowledge and Data Engineering* **13** (1): 64–78, doi:10.1109/69.908981 .
8. <sup>^</sup> Andoni, Alexandr; Indyk, Piotr (2008), "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions", *Communications of the ACM* **51** (1): 117–122, doi:10.1145/1327452.1327494 .
9. <sup>^</sup> Zadeh, Reza; Goel, Ashish (2012), *Dimension Independent Similarity Computation*, arXiv:1206.2082 .
10. <sup>^</sup> Henzinger, Monika (2006), "Finding near-duplicate web pages: a large-scale evaluation of algorithms"  (PDF), *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. ACM*.
11. <sup>^</sup> Charikar, Moses S. (2002), "Similarity estimation techniques from rounding algorithms", *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing. ACM*.
12. <sup>^</sup> Gurmeet Singh, Manku; Jain, Arvind; Das Sarma, Anish (2007), "Detecting near-duplicates for web crawling", *Proceedings of the 16th international conference on World Wide Web. ACM*.
13. <sup>^</sup> Das, Abhinandan S. et al. (2007), "Google news personalization: scalable online collaborative filtering", *Proceedings of the 16th international conference on World Wide Web. ACM*.
14. <sup>^</sup> R. R. Salakhutdinov and G. E. Hinton. Semantic hashing. In SIGIR workshop on Information Retrieval and applications of Graphical Models, 2007.
15. <sup>^</sup> Weiss, Yair, Antonio Torralba, and Rob Fergus. "Spectral hashing." *Advances in neural information processing systems*. 2009.
16. <sup>^</sup> Heo, Jae-Pil, et al. "Spherical hashing." *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. IEEE*, 2012.
17. <sup>^</sup> Dean, Thomas, et al. "Fast, accurate detection of 100,000 object classes on a single machine." *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on. IEEE*, 2013.

Categories: [Hash functions](#) | [Clustering criteria](#) | [Hashing](#) | [Probabilistic data structures](#)

This page was last modified on 7 August 2015, at 00:07.

Text is available under the [Creative Commons Attribution-ShareAlike License](#); additional terms may apply. By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#). Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](#), a non-profit organization.

[Privacy policy](#) [About Wikipedia](#) [Disclaimers](#) [Contact Wikipedia](#) [Developers](#) [Mobile view](#)

