Understanding MinHash: The Intuitive Guide

Finding Similar Items in Large Datasets

Prof. Georgina Cosma

March 6, 2025

The Problem: Finding Similar Items

Real-World Challenge

- Web companies need to find similar documents.
- Search engines need to remove duplicate results.
- Recommendation systems need to group similar items.
- NLP models should avoid duplicate training examples.

But It's Hard Because...

- **Scale:** Comparing billions of documents.
- **Complexity:** Documents can be partially similar.
- **Efficiency:** Can't compare everything to everything.

Analogy: Imagine you're a librarian with millions of books. How do you quickly find books with similar content without reading every single one?

Document Similarity: The Basics

Document 1	Document 2
"The quick brown fox jumps over the lazy dog"	"The brown fox jumps over the sleepy dog"

How do we measure similarity?

- Break into sets of words (or sequences of words).
- Compare how much the sets overlap.
- Use Jaccard similarity:

$$\text{Jaccard Similarity} = \frac{|A \cap B|}{|A \cup B|}$$

Example:

Doc 1 words: {the, quick, brown, fox, jumps, over, lazy, dog}

Doc 2 words: {the, brown, fox, jumps, over, sleepy, dog}

Jaccard: $\frac{6}{9} \approx 0.67$ (6 words in common, 9 unique words total)

The Challenge of Scale

Simple but Inefficient Approach

1. Take every pair of documents.
2. Compute their Jaccard similarity.
3. Keep pairs with similarity above some threshold.

The Problem

If you have 1 million documents:

- Number of possible pairs: $\binom{1{,}000{,}000}{2} \approx 5 \times 10^{11}$ (500 billion).
- Comparing sets is expensive.
- This would take months or years to compute!

We need a shortcut!

Enter MinHash: The Clever Shortcut

The Big Idea

- Create a small "signature" for each document.
- Signatures preserve similarity information.
- Comparing signatures is much faster than comparing documents.
- Only need to closely examine documents with similar signatures.

The Magic Insight

If we choose the minimum hash value from each document, the probability they match equals their Jaccard similarity!

Analogy: MinHash is like creating a fingerprint for each document. Instead of comparing entire documents, we compare their fingerprints, which is much faster!

How MinHash Works: Step 1

Converting Documents to Sets

Break documents into "shingles" (overlapping sequences of words or characters).

Original text: "The quick brown fox"
2-word shingles:

- "The quick"
- "quick brown"
- "brown fox"

Each document becomes a set of these shingles.

Why Shingles? Shingles capture the structure and meaning of the text, making it easier to detect similarities.

How MinHash Works: Step 2

Apply Hash Functions

Apply multiple different hash functions to each shingle.

Shingle	**Hash 1**	**Hash 2**	**Hash 3**
"The quick"	142857	107364	635829
"quick brown"	293847	548372	881234
"brown fox"	912345	320987	156789

Note: Real hash values are typically 32-bit or 64-bit integers.

How MinHash Works: Step 3

Keep Only the Minimum Hash Values

For each hash function, keep only the smallest value from all shingles.

Shingle	**Hash 1**	**Hash 2**	**Hash 3**
"The quick"	**142857**	**107364**	635829
"quick brown"	293847	548372	881234
"brown fox"	912345	320987	**156789**
Minimum	**142857**	**107364**	**156789**

The document's MinHash signature becomes [142857, 107364, 156789].

Why This Works: The Key Insight

The Mathematical Magic

- For any hash function, the probability that two sets have the same minimum hash value equals their Jaccard similarity.
- If documents share 70% of their shingles, they'll have the same minimum hash about 70% of the time.
- Using multiple hash functions gives us a more reliable estimate.

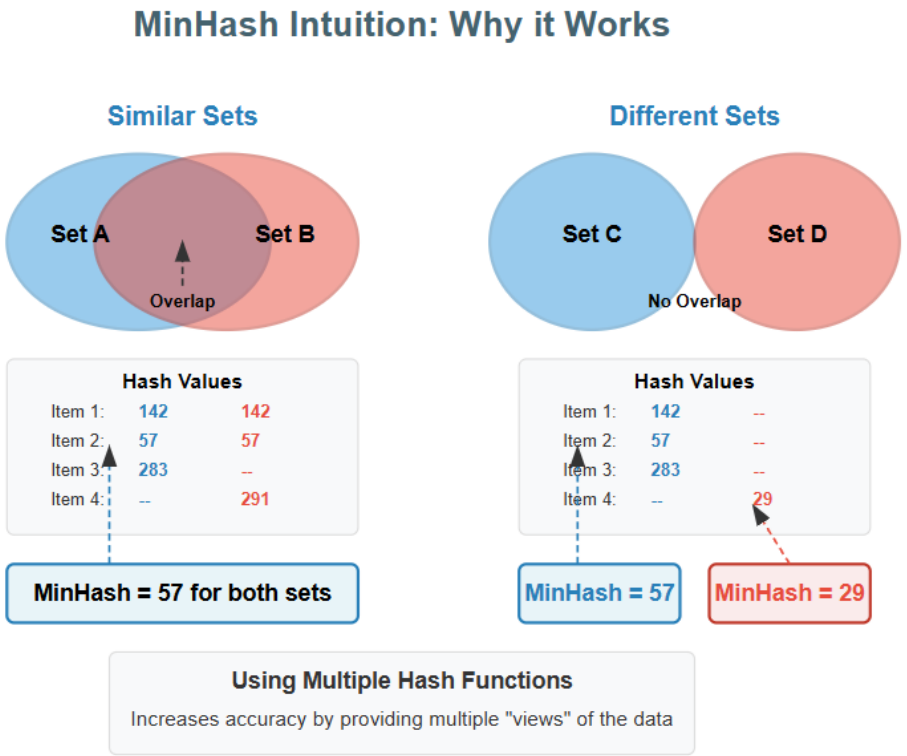
Think of it this way:

If you randomly pick an element from two sets:

- The chance it's in both sets = Jaccard similarity.
- MinHash is like randomly picking the element with the smallest hash value.

Visual Explanation: Why MinHash Works

Let's visualize with a simplified example:



- If sets are very similar, they'll tend to have the same minimum element.
- If sets have little overlap, they'll likely have different minimum elements.
- Using multiple hash functions gives us multiple "views" of the data.

Reading the MinHash Intuition Diagram

Step 1: Understand the Sets

- Colored ellipses represent sets. Overlapping regions show shared elements.
- Left side: similar sets with significant overlap.
- Right side: different sets with no overlap.

Step 2: Examine the Hash Values

- Each table shows hash values for elements in the sets.
- Blue numbers belong to the left set, red numbers to the right set.
- Dashes (–) indicate the element isn't present in that set.

Step 3: Identify and Compare MinHash Values

- MinHash is the smallest hash value in each set(arrows point to these).
- Similar sets (left) have the same MinHash value (57).
- Different sets (right) have different MinHash values (57 vs 29).

Practical Example: Two Similar Documents

Now let's work through a complete practical example:

Document A

"The quick brown fox jumps over the lazy dog"
Shingles:

- "The quick"
- "quick brown"
- "brown fox"
- "fox jumps"
- "jumps over"
- "over the"
- "the lazy"
- "lazy dog"

Document B

"The quick brown fox leaps over the sleepy dog"
Shingles:

- "The quick"
- "quick brown"
- "brown fox"
- "fox leaps"
- "leaps over"
- "over the"
- "the sleepy"
- "sleepy dog"

MinHash Signatures: Comparing Documents

Document A Signature

- Hash 1: 24
- Hash 2: 56
- Hash 3: 18
- Hash 4: 92
- Hash 5: 41

Document B Signature

- Hash 1: 24
- Hash 2: 56
- Hash 3: 18
- Hash 4: 77
- Hash 5: 63

Similarity Estimate

- 3 out of 5 hash values match.
- Estimated Jaccard similarity: $3/5 = 0.6$ (60%).
- Actual Jaccard similarity: $5/11 = 0.45$ (5 common shingles out of 11 unique shingles).
- We get a reasonable approximation with much less computation!

Scaling Up: Locality-Sensitive Hashing (LSH)

Still Too Many Comparisons?

MinHash helps, but comparing all signature pairs is still $O(n^2)$.

Solution: LSH "Banding Technique"

- Divide MinHash signature into bands.
- Hash each band to a bucket.
- Only compare documents that share at least one bucket.
- Dramatically reduces the number of needed comparisons.

Example: Banding MinHash Signatures from Our Documents

Band	Document	Hash 1	Hash 2	Hash 3	Bucket
Band 1	Doc A	24	56	18	Bucket A
	Doc B	24	56	18	Bucket A
Band 2	Doc A	92	41	–	Bucket B
	Doc B	77	63	–	Bucket C

Practical Applications of MinHash

Real-World Uses

- **Web Search**: Google uses similar techniques to remove duplicate web pages.
- **Plagiarism Detection**: Finding copied content across documents.
- **Image Similarity**: Can be adapted to find similar images.
- **Recommendation Systems**: Finding similar products or content.
- **Language Models**: Deduplicating training data (as in the paper we discussed).

Key Advantages

- Works on massive datasets (billions of items).
- Handles partial similarity, not just exact matches.
- Much faster than direct comparison.
- Can be distributed across multiple computers.

Implementation Considerations

Practical Tips

- **Number of hash functions**: More functions = more accurate but slower.
- **Shingle size**: Larger shingles catch more specific similarities.
- **LSH bands**: More bands catch more similar pairs but with more false positives.
- **Hash functions**: Must be independent and uniformly distributed.

Common Implementations

- Python: `datasketch` library.
- Java: `MinHash` in `Apache DataSketches`.
- Production systems often use custom implementations.

Recap: Why MinHash is Brilliant

1. **Converts the similarity problem** from comparing sets to comparing small signatures.
2. **Mathematical guarantee**: Probability of matching min-hash values equals Jaccard similarity.
3. **Scalable**: Can handle billions of documents efficiently.
4. **Adaptable**: Works for text, images, or any data that can be represented as sets.

Remember the Key Insight

By keeping only the minimum hash values, we create a compact document signature that preserves similarity information while being much faster to compare.

Questions?

Thank You!

Any questions or examples you'd like to explore?