

Portfolio Question 8.2.P - MinHash

Daniel Lawson University of Bristol

Portfolio 8.1.P (v4.0.0)

Portfolio Question 8.2.P - Hashing big data

- ▶ Imagine you are training an LLM and wanting to keep track of the documents its seen already, to avoid duplication and allow you to downweight topics by the proportion of computer generated content they contain.
 - ▶ How does the MinHash algorithm help solve this problem?
 - ▶ How would a sketch help?
 - ▶ Discuss the approach and any competition for the LLM problem.

MinHash motivation

- ▶ Consider a very large, potentially sparse, **binary** feature space for which we have observations $A = \{x_i\}$ and $B = \{x_k\}$. How similar are they?
- ▶ One natural measure is the **Jaccard Similarity**:

$$J(x_i, x_j) = \frac{x_i \cap x_j}{x_i \cup x_j}$$

- ▶ This is slow to compute with a large sparse features space, such as **words**.
- ▶ The solution is to approximate the similarity via MinHash.

MinHash algorithm

- ▶ To compute a single MinHash Signature:
 - ▶ Use a **random hash function** and apply it to all values in A and B .
 - ▶ Compute the minimum of each of these.
 - ▶ The probability of these being equal turns out to $J(A, B)$.
- ▶ To estimate J , we simply do this several times.
- ▶ This was used for website Duplicate detection by AltaVista and was confirmed to be still in use by Google in 2007. There were a lot of websites, even then.
- ▶ See e.g. Chris McCormick's **Minhash tutorial** or the **Mining of Massive Datasets** book and course.

Sketching

- ▶ Sketching is obtaining the frequency properties of your data from a data stream.
- ▶ One important class is probabilistic counting, which addresses how many of each class there are.

Count-min-sketch

- ▶ Count-min-sketch works just like a bloom filter, except that we store an integer for each has rather than a single bit.
- ▶ We initialise $\mathbf{b}_r = \mathbf{0}$, and then:
 1. Compute k hashes in $(0, r]$, $h_k(x_i)$
 2. Add one to all bits hashed into, i.e. $b_r(h_k(x_i)) + = 1$
- ▶ At lookup time, the number of items is estimated to be

$$\operatorname{argmin}_{h_k(x_i)} b_r(h_k(x_i))$$

i.e. the minimum count.

- ▶ See e.g. Python implementation of **Count Min Sketch** by Rafael Carrascosa (part of PyPI)

Sketching Example

5														
147														
101														
86	→	$h_1(86) = 4$	h_1	0	0	1	0	1 ⁻¹	0	0	0	0	0	1
14		$h_2(86) = 2$	h_2	1	0	0 ⁻¹	0	0	1	0	0	1	0	
⋮		$h_3(86) = 9$	h_3	0	0	0	1	0	0	0	1	1	0 ⁻¹	
		$h_4(86) = 1$	h_4	0	0 ⁻¹	0	0	2	0	0	1	0	0	

Discussion

- ▶ MinHash allows rapid recall of whether we've seen data before.
- ▶ Sketching allows us to count (classes) of data.
 - ▶ We need a “computer generated content” detector for this to work.
- ▶ Conceptually they together allow us to compute a weight w_i for data:
 - ▶ MinHash a new datapoint: is it new?
 - ▶ Estimate the frequency of it by adding it to the sketch
- ▶ It is not clear this is the best way to do this
 - ▶ At what scale of data is this approach sensible?
 - ▶ These are classical algorithms, are new ones better?
 - ▶ What do people do in practice?