Content-based Document Routing and Index Partitioning for Scalable Similarity-based Searches in a Large Corpus

Deepavali Bhagwat, Kave Eshghi, Pankaj Mehra

Overview

- http://videolectures.net/kdd07_bhagwat_cbdr/
- Goal: Similarity search
- Method:
  - Partition feature index and spread it across servers
  - Each doc in the index is represented on a few servers
  - A query doc is only processed on a few servers w/ high probability of representing existing similar docs.
- Claims:
  - Scales well
  - Minimal impact on precision and recall

Their big observation:

Any algorithm that creates probabilistic measures of document similarity (ex. TTTD chunking or any chunking, really) can also be used to decide how to partition documents to servers.

How to route

1. Make a list of document features: $H(f_n)$
2. Select m of them: $bot_m(H(f_n))$
3. Select partitions: $R = bot_m(H(f_n)) \bmod k$

   (slight chance of collisions? A document might be sent to fewer partitions.)
4. Copy $H(f_n)$ -> f_n (yes, all of it) into to the inverted keyword index for every partition in R (yes, all of them).

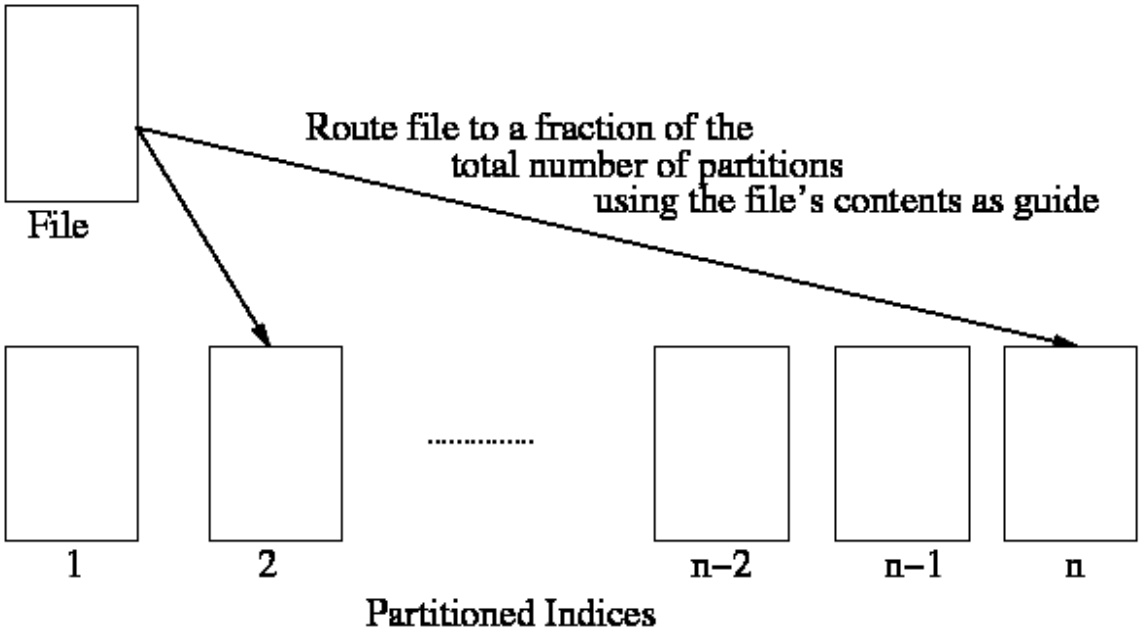

Figure 1: Our solution for document routing

Some math

- Basically, all the same math that says that things like TTTD and minhash and stuff like that works says that this works.

THEOREM 1. *Consider two sets S_1 and S_2, with $H(S_1)$ and $H(S_2)$ being the corresponding sets of the hashes of the elements of $S1$ and $S2$ respectively, where H is chosen uniformly and at random from a min-wise independent family of permutations. Let $min(S)$ denote the smallest element of the set of integers S.*

$$P(min(H(S_1)) = min(H(S_2))) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

Some math

LEMMA 1. *Let S_1 and S_2 be two sets. Let $I = |S_1 \cap S_2|$ and $U = |S_1 \cup S_2|$. Let $B_1 = bot_m(H(S_1))$ and $B_2 = bot_m(H(S_2))$, where H is a min-wise independent hash function. Then*

$$P(B_1 \cap B_2 = \emptyset) \leq \frac{(U-I)(U-I-1)\ldots(U-I-m-1)}{U(U-1)\ldots(U-m-1)}$$

Let $s = I/U$, i.e. s is the Jaccard similarity measure between S_1 and S_2. A good approximation of the above, when m is small and U is large, is

$$P(B_1 \cap B_2 = \emptyset) \leq (1-s)^m + \epsilon$$

ϵ is a small error factor in the order of $1/U$.

Some math

COROLLARY 1. *Let f_1 and f_2 be two documents with similarity measure s. When they are each routed to m partitions using the algorithm above, the probability that there will be at least one partition to which both of them are routed is at least* $1 - (1 - s)^m$

Results

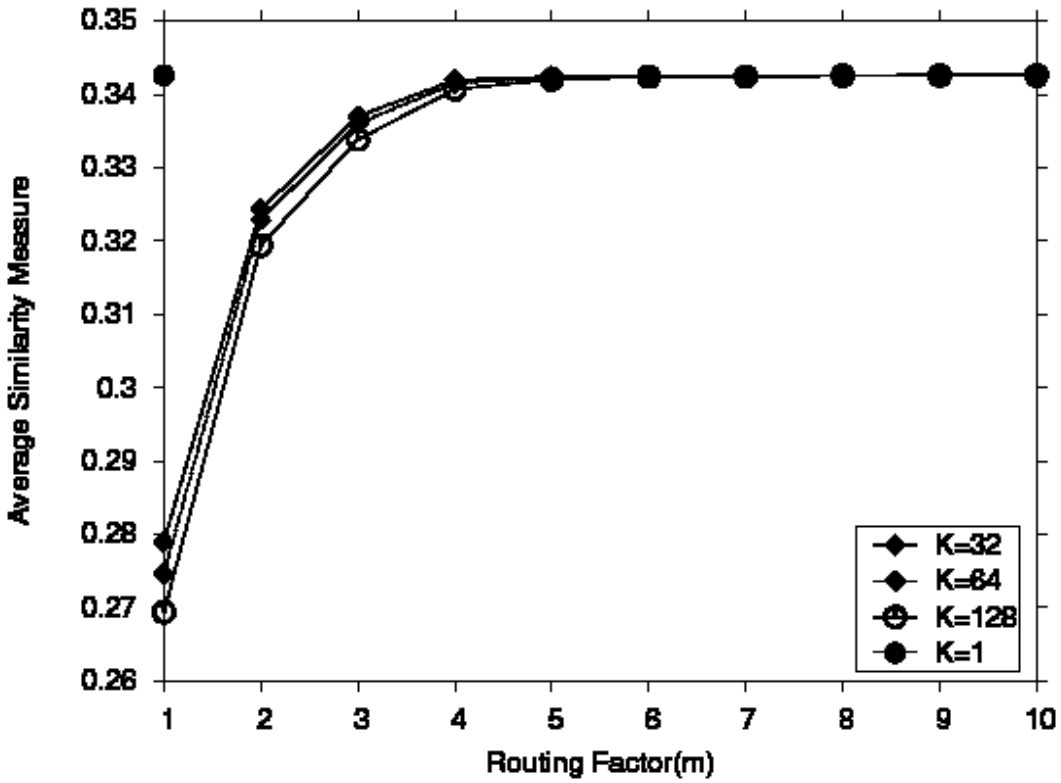

Figure 4: Effect of the routing factor on the average similarity measure

Results

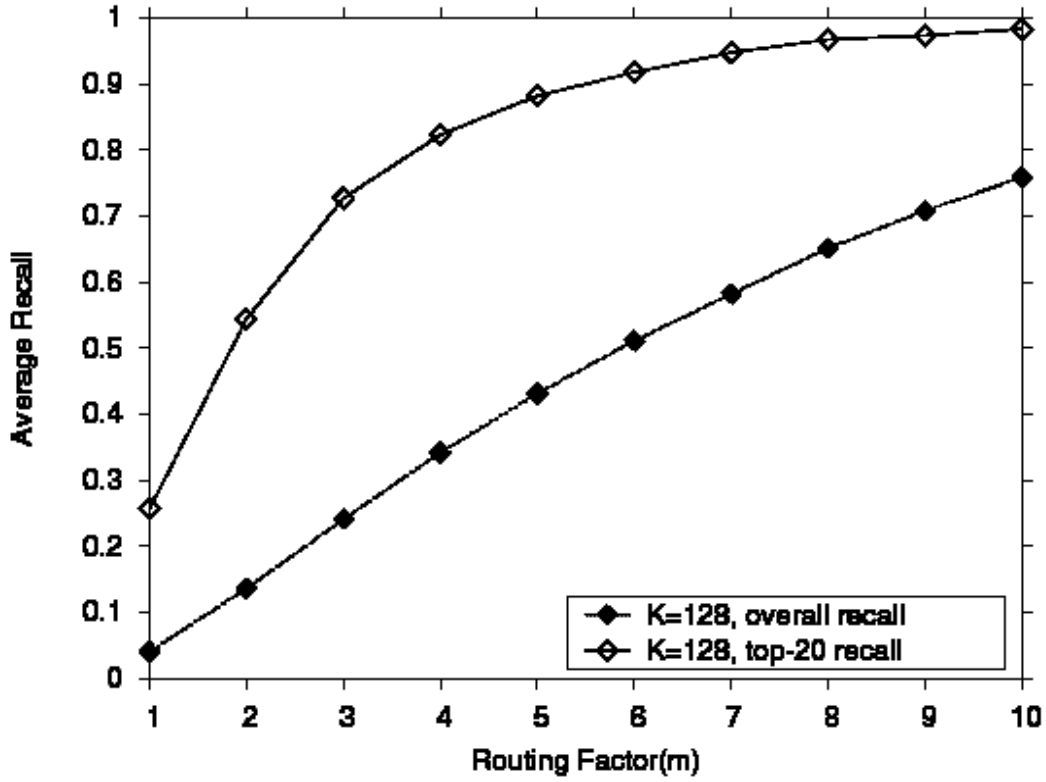

Figure 5: Effect of the number of partitions on the overall recall

Results

Actual quote from the paper:

> “However, if an application requires that every similar document to a query be found, then one can easily adopt a policy of querying each and every partition instead of just m out of K. Such a scheme will preserve the original recall of every query as was the case when there existed just one index(K = 1).”

...

Results

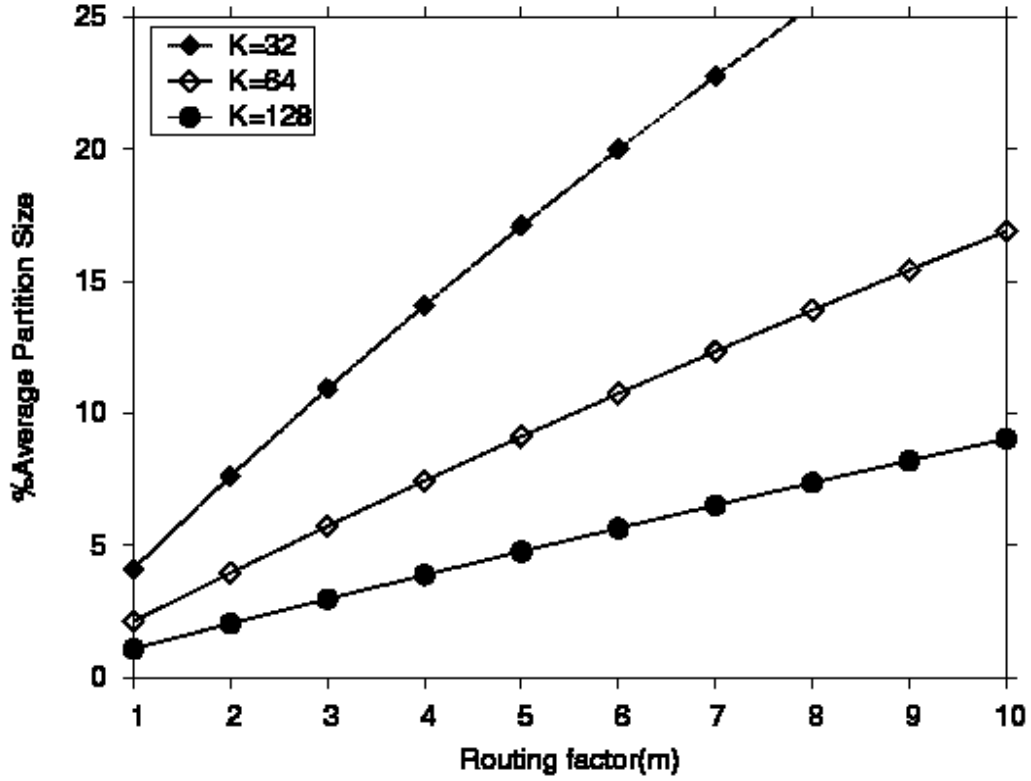

Figure 7: %Average partition sizes for increasing values of the routing factor

Issues not really addressed in paper

- Duplication: New index is ~mx larger than a single index even if “% average partition size” is smaller.
- Stateless & not dependent on data: Load balancing issues
- “Future work will consist of quantifying our losses in the form of storage space and finding out if our gain, in the form of better bandwidth utilization and throughput, outweighs this loss.”