Indexing inexact proximity search with distance regression in pivot space

Ole Edsberg Magnus Lie Hetland

Department of Computer and Information Science
Norwegian University of Science and Technology

Similarity Search and Applications, 2010

The problem

Our task: Speed up proximity search in cases where:

- Distance calculation is expensive.
- Distance-based indexing is needed, because the contents of the data objects cannot be used in the index.
- Some search inexactness is acceptable, meaning we are allowed to trade som search accuracy in return for reduced search computation cost.

Our contribution: A new indexing scheme that in some cases provides better computation / accuracy trade-offs than the competition. It also has some draw-backs.

Related work

E. Chávez, K. Figueroa, and G. Navarro.
Effective proximity retrieval by ordering permutations.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 30(9):1647–1658, 2008.

Takeaway:

- How to perform inexact search by ordering the database according to *promise value* function.
- A specific promise value function, which we use as a baseline in our experiments.
- Experimental setup.

Pivot space

Pivot set:

$$\mathbb{P} = (p_1, p_2, \ldots, p_m)$$

Mapping object o to pivot space.

$$\Phi(o) = (d(o, p_1), d(o, p_2), \ldots, d(o, p_m))$$

The baseline: Permutation based promise values

The promise value for indexed object u with respect to query q is the correlation (rank correlation coefficient, Spearman's ρ) between the ordering permutation of $\Phi(u)$ and that of $\Phi(q)$.

What makes a good promise value function?

Two ideas for promise value functions

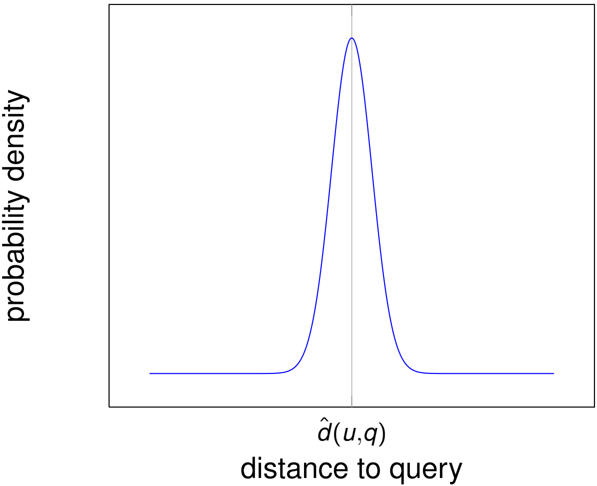
Distance estimate

$$\hat{d}(u, q)$$

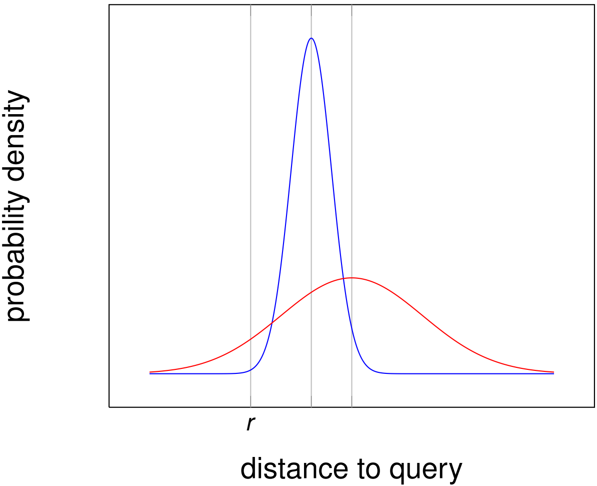
Probability of inclusion

$$\Pr(d(u, q) \leq r)$$

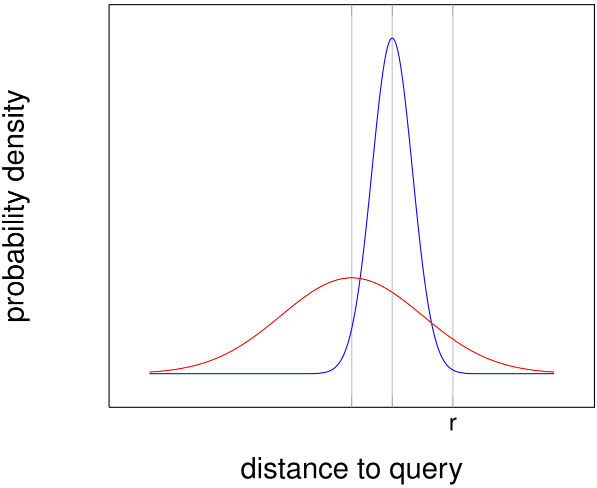
Uncertainty in distance estimates



Should red or blue object be visited first?



Should red or blue object be visited first?



Linear model of distance for indexed object u

$$d(u,q) = \beta_{\langle u,0\rangle} + \sum_{i=1}^{m} \beta_{\langle u,i\rangle} d(q,p_i) + \epsilon_u \,,$$

Regression-based index (one model per object!)

With n objects to index and m pivots, an $n \times (m+1)$ matrix:

$$\begin{pmatrix} \hat{\beta}_{\langle u_1,0\rangle}, \hat{\beta}_{\langle u_1,1\rangle}, \ldots, \hat{\beta}_{\langle u_1,m\rangle} \\ \hat{\beta}_{\langle u_2,0\rangle}, \hat{\beta}_{\langle u_2,1\rangle}, \ldots, \hat{\beta}_{\langle u_2,m\rangle} \\ \ldots \\ \hat{\beta}_{\langle u_n,0\rangle}, \hat{\beta}_{\langle u_n,1\rangle}, \ldots, \hat{\beta}_{\langle u_n,m\rangle} \end{pmatrix}$$

- The coefficients can be discretized to save space.
- Plus $2n+2$ additional values if we want probabilities.

Building the index

1. Select n' *training queries* from the objects to be indexed.
2. For each training query q', calculate $\Phi(q')$.
3. For each object to be indexed u:
   3.1 For each training query q', calculate $d(u, q')$.
   3.2 Solve the least squares linear regression problem to find the $m + 1$ coefficients β_u.
   3.3 Store the coefficients in the index.
   3.4 If we want probabilities, also store $\hat{\sigma}_u$, the estimated standard deviation of ϵ_u:

$$\hat{\sigma}_u = \sqrt{\frac{\sum_{i=1}^{n'} (d(u, q'_i) - \hat{d}(u, q'_i))^2}{n' - m - 1}}$$

4. If we want probabilities for the k-NN queries, also store the estimated search radius for each k.

(Detail glossed over in this presentation: we exclude u from the training queries used to fit its own model.)

Distance estimates as promise values

$$\hat{d}(u,q) = \hat{\beta}_{\langle u,0\rangle} + \sum_{i=1}^{m} \hat{\beta}_{\langle u,i\rangle} d(q,p_i)$$

Probability-based promise values

$$\frac{r - \hat{d}(u, q)}{\hat{\sigma}_u}$$

- Depends on a lot of assumptions.
- We also ignore the consequence of excluding *u* from its own training queries.

Storage costs

With n objects to index and m pivots,

- For distance estimates: $n(m+1)$ coefficients. (Can be discretized at the cost of some accuracy.)
  - For probabilities: $2n+2$ additional values.
- Permutation-based index: $nm\lceil\log_2(m)\rceil$ bits in total.

Index building costs

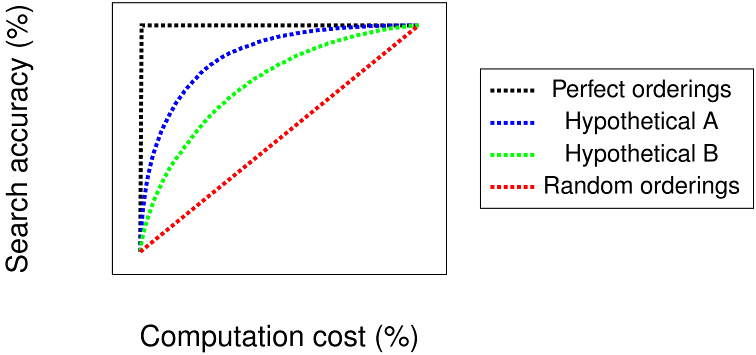
With n objects to index, m pivots and n' training queries,

- Regression-based scheme: $n'(n+m)$ distance calculations plus the solution of n linear regression problems.
- Permutation-based scheme: nm distance calculations, plus some sorting.

Experimental setup

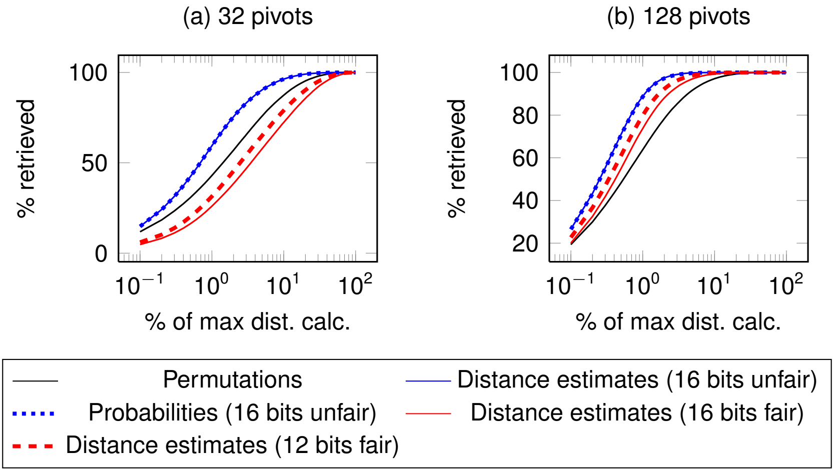
- We borrowed the experimental setup from Chávez, Figueroa & Navarro's evaluation of the permutation-based scheme.
- Pivots selected randomly.
- Also evaluated versions with pivot set reduced to make storage cost equal to permutation-based index.
- Both synthetic and real-world data sets, but results on real-world data may have more validity.

Evaluating promise value functions: computation / accuracy trade-offs

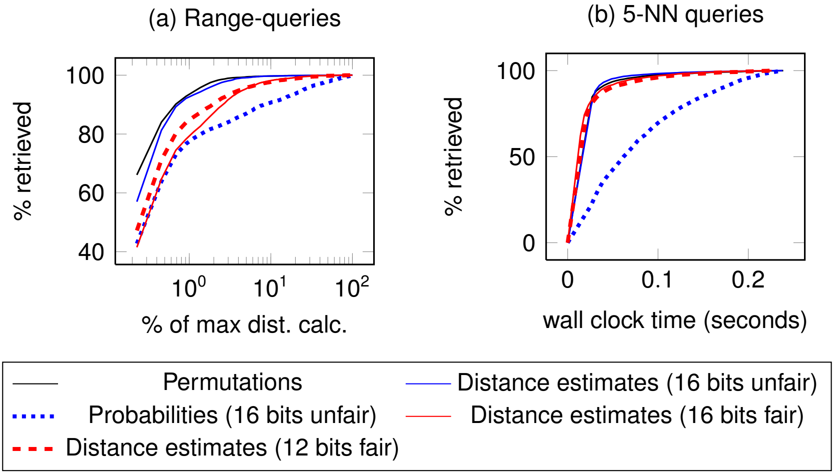


(Average over many queries.)

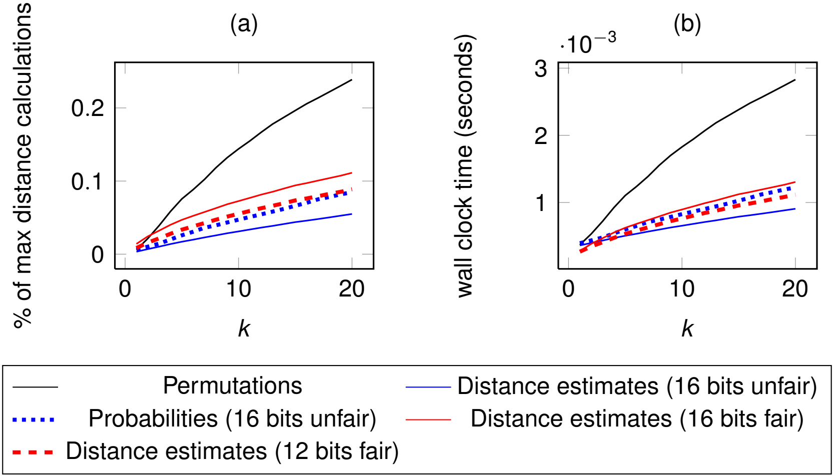
Results on normalized edit distance (NED)



Results on documents (TREC)



Results on face images (FERET)



Why were the probability-based promise values sometimes worse, and never better, than the distance estimates?

Conclusion

Regression-based scheme show some promise, but:

- Takes a lot of time to build the index.
- Vulnerable to deviation from assumptions.