

Outline of Inverted List Heuristics for kNN

*Based on notes by David D. Lewis, with some editing by Vldimir Menkov
July 20, 2003*

From daviddlewis@worldnet.att.net Sun Jul 20 16:24:09 2003

Efficiency for kNN is poorly understood, so we shouldn't aspire to a highly tuned combination of inverted list heuristics (ILHs). Our experiments should instead get basic results on the value of different ILHs in isolation, and present these basic results as a point of comparison with the randomization results.

I see the techniques listed below as the basic ILHs. This list is a revision of the one in my Dec 2002 writeup.

Terminology: In the following I use the term "impact" to mean "within document term weight", following Ahn & Moffat's usage. I use "training impact" to mean impact of a term in a training document, and "test impact" for impact of a term in a test document. I use AQ for "approximation quality", a subject discussed at length in an accompanying message.

ILH 1. Discard low impact terms from each training document before/during inverted file construction

Choice 1a. For each training document, keep terms w/:

- i. Top X impacts
- ii. Top X% of impacts in doc
- iii. Summed impact = X% of doc's total
- iv. Impact $\geq$ X
- v. Overall impact guaranteeing AQ level Y

Choice 1b.

- i. Length normalize before discarding
- ii. Length normalize after discarding

DESCRIPTION: Reduce size of inverted file by reducing size of each document. We discard terms that on average will have little impact on document scores. This is the most ancient technique for speeding up inverted file retrieval. It was the original motivation for IDF weighting.

ADVANTAGES: Easier to guarantee per-document AQ measures.

DISADVANTGES: Not sensitive to test impacts.

NOTES

1. Discarding low impact terms from *test* document is covered under inverted list pruning (ILH 2).

ILH 2. Reading only selected inverted lists ("inverted list pruning")

Choice 2a. Read lists in order by

- i. List length (shortest first)
- ii. IDF of term (same as i. unless auxiliary data used to get IDF)
- iii. Test doc within-doc wt of term
- iv. Max impact (training doc within-doc wt) on list
- v. Min impact on list
- vi. Average impact on list
- vii. Max contribution (test doc term wt times max impact)
- viii. Min contribution
- ix. Average contribution

Choice 2b. Stop reading lists when

- i. X lists have been read
- ii. X% of lists have been read
- iii. X% of sum of length of lists has been read
- iv. AQ level X provably reached
- v. Allocated compute time is used up

Choice 2c. Renormalize length of test document using only selected terms

- i. Yes
- ii. No

DESCRIPTION: Some inverted lists contain mostly low impact terms making little contribution to scores. Due to IDF, these tend to be longer lists as well, so avoiding processing them can speed things up a lot.

NOTES

1. I listed a lot of possibilities in 2a, but I think only 2a.i., 2a.iii., 2a.vii., and 2a.ix are worth trying.

2. Should training document scores reflect the original length (e.g. 2-norm) of the test document? Or the length ignoring the pruned inverted lists? If unweighted k-NN is used it doesn't matter, but for weighted k-NN the question is more tricky.

For weighted kNN the question is more tricky. If one views pruning as a kind of classification-time feature selection (analogous to ILH 1 for training documents), that suggests renormalizing. If one views it as a way of approximately computing the original similarities, then instead just using the approximation (or computing the full similarity for selected docs, i.e. ILH ?) makes more sense.

3. BL85 is one version of this method.

ILH 3. Use all lists, but read only the high-impact portion

of each list

Choice 3a. Read these entries only

- i. Top X
- ii. Top X%
- iii. With impact $\geq X$ (fixed threshold)
- iv. With impact $\geq X * \text{highest doc score}$ [Persin, 1996]
- v. Sufficient to guarantee AQ X

Choice 3b. Order to read lists in
[See 2a.]

DESCRIPTION: Instead of sorting inverted list entries by doc ID, we sort them by impact. We access all lists, one at a time, but read each list only until some stopping criterion is met for that list, then go on.

NOTES:

1. Choice 3b is relevant only for stopping criteria (3a.iv and some 3a.v) which are affected by order of lists.
2. There are fancier methods which make several passes over the list, but those don't seem worth looking at for now.

ILH 4. Speeding up finding top scores

DESCRIPTION: The simplest way to get the k neighbors is an $O(n \log n)$ sort of the accumulators. Faster techniques can do this in slightly over $O(n)$. A simple one is to use a heap of size k, and a threshold equal to the kth highest current score. These techniques aren't really "heuristics", since they find the exact answer (based on the possibly approximate scores).

ILH 5. Computing exact scores for neighbors

Choice 5a.

1. Do it
2. Don't do it

DESCRIPTION: Once a set of neighbors has been selected using approximate scores, it may (or may not) be desirable to go back and compute the exact scores of these neighbors, using their full training document vectors and the full test document vector. This requires keeping the training document vectors in memory - a nontrivial cost. This technique is needed with the randomization methods, since they don't return scores that are useful for weighted kNN

methods, but it's not clear this is useful with inverted
file heuristics.
