LSH BANDING FOR LARGE-SCALE RETRIEVAL WITH MEMORY AND RECALL CONSTRAINTS

Michele Covell and Shumeet Baluja

Google Research, Google Inc.
1600 Amphitheatre Parkway, Mountain View CA 94043

ABSTRACT

Locality Sensitive Hashing (LSH) is widely used for efficient retrieval of candidate matches in very large audio, video, and image systems. However, extremely large reference databases necessitate a guaranteed limit on the memory used by the table lookup itself, no matter how the entries crowd different parts of the signature space, a guarantee that LSH does not give. In this paper, we provide such guaranteed limits, primarily through the design of the LSH bands. When combined with data-adaptive bin splitting (needed on only 0.04% of the occupied bins) this approach provides the required guarantee on memory usage. At the same time, it avoids the reduced recall that more extensive use of bin splitting would give.

Index Terms— Multimedia databases, Information retrieval, Fingerprint identification, Pattern matching

1. INTRODUCTION

Many approximate nearest-neighbor problems require candidate retrieval under tight memory and computation constraints [1, 2, 3]. Most notable among these are audio and video retrieval systems that must handle both time-based editing (excerpts and mash-ups) and frame-based editing (visual overlays and audio re-mixes). For partial matches across time, the size of the database must grow according to the number of minutes of content, instead of just according to the number of distinct clips, leading to billions of entries. Due to the large size of the reference set, we need to know that we will never be required to read back more than a small fraction of that set, no matter how these entries crowd different parts of the signature space.

Data-adaptive tree structures, such as spill trees [4], can be used to limit the maximum number of retrieved candidates, using the variable depth to subdivide crowded portions of the space. However, the recall rate for such adaptive tree approaches is difficult to monitor, since the radius of retrieval in each region is not known before construction of the tree. For applications like partial-copy detection, where high and predictable recall rates are needed, we propose using data-adaptive approaches only when even careful design of non-adaptive tables fail to meet our memory limits.

Locality Sensitive Hashing (LSH) [5] provides predictable recall rates, since the retrieval characteristics are defined by the off-line design of the hash keys. With LSH, each reference entry is inserted into L tables using different subspace projections. The simplest and most computationally efficient projections are axis parallel: from a BL-dimensional signature, B distinct dimensions are taken for each of L LSH "bands". Our work uses axis-parallel projection, since it supports our discrete-valued, non-ordinal signatures [2, 6].

We propose to use LSH, with off-line optimization of the hash subspaces, in order to minimize the worst-case expected crowding. We also use (on-line) data-adaptive bin splitting, as required by the actual reference database and our memory guarantees. Most of this work focuses on the selection of the hash subspace to limit the expected, worst-case crowding in the LSH tables. While axis-parallel projection limits the set of possible distinct hash subspaces to $\prod_{l=0}^{L-2} \binom{B(L-l)-1}{B-1}$, that number is still unmanageably large for our operating point [2], giving $1.9\text{x}10^{98}$ distinct 25x4 possible groupings. The size of this space leads us to use a greedy solution.

In a closely related problem [1], a greedy approach was also taken to select among the hash-subspace possibilities. In that work, the reward criteria was minimum within-band mutual information, thereby maximizing the number of bits of entropy across the group of hash tables and minimizing the average candidate-list length. As will be seen in Figure 3, minimizing mutual information does not minimize the worst-case memory usage for a look up.

To minimize this *worst-case* cost, our reward function focuses on providing the smallest-size, and smallest number of, overfull LSH bins. We select a *submodular* reward function, based solely on these overfull bins. This submodular property both improves the speed of each search for groupings (effectively allowing additional search without prohibitive training time) and provides a bound on how far below optimal our solution is. By bounding the optimum solution, we know how much loss we suffer from any proposed solution. This bound on the optimal, taken from the greedy solution, can be combined with "seeded" greedy search, providing both efficient search into improvements on the fully greedy solution and a stopping criteria on an outer search for a good seed.

Section 2 reviews submodularity, how this property allows efficient evaluation of greedy solutions, and the bound provided by the submodularity. Section 3 reviews the specifics of our problem and introduces our submodular reward function. Section 4 introduces an efficient approach to representing the above-threshold portions of our LSH frequency distributions, which is needed to support efficient evaluation on this problem. Section 5 discusses seeded greedy search, to improve on our basic greedy solution. We provide experimental results in Section 6, using data-adaptive bin splitting to handle any remaining occupancy violations. Section 7 concludes and proposes future directions for study.

2. SUBMODULAR REWARD BOUNDS AND EFFICIENT GREEDY SEARCH

Since the optimal solution to our grouping problem is *NP*-hard, we take a greedy approach to finding a "good" solution. For general reward functions, even finding the greedy solution requires $O(n^2)$ evaluations of that reward function (with $n = LB$ for our problem). In this section, we review submodular-reward functions, which provide an efficient way to find the same greedy solution that would have been found by the $O(n^2)$ evaluation but uses only $O(n)$ evaluations and, more importantly, which provide bounds on how far from optimal a greedy solution can be.

Submodular reward functions have been proposed since 2004 for greedy solutions [7]. A submodular reward function, $R()$ compares the reward of adding an element C to possible solution sets A and B, where $A \subseteq B$. For a submodular reward function, the incremental reward for adding C to solution B is always less than or equal to the incremental reward for adding C to solution A: $R(B \cup C) - R(B) \leq R(A \cup C) - R(A)$. If we use a greedy algorithm to find the solution to a problem with submodular reward structure, we are guaranteed to be within $(1 - 1/e)$ of the reward of optimal solution [7] (which is *NP*-hard to find).

Furthermore, as Leskovec, et al. [8] describe, we can find the greedy solution more efficiently than is possible with more general reward structures by using this decreasing incremental-reward characteristic to sort our solutions, without full re-evaluation at each step. A version of dynamic programming, Cost-Effective Lazy Forward (*CELF*) evaluation maintains a list of available candidate steps, sorted based on the recorded incremental reward of that step. This sort mixes together current and out-of-date incremental reward values. If the best next step has an out-of-date reward, its incremental reward is re-evaluated and it is bubble sorted down the list. If the best next step has a current reward, it is guaranteed to be the best step for a greedy approach. We take that candidate step (removing it from the available list) and update the remaining candidates. Candidates that would have used the just-taken dimension are removed. If the just-updated LSH band is full, remaining candidates for that band are removed from the list; otherwise, their incremental rewards are marked as out-of-date. In this way, many steps can be taken towards the greedy solution without re-evaluation of the full set of possible rewards. Since we use a submodular reward, we can sort together a mixture of out-of-date and current incremental reward values and know that, when the best candidate addition has a current reward, it will also be the best candidate addition over all current possibilities.

3. SUBMODULAR REWARD FUNCTION FOR MINIMIZING ABOVE-THRESHOLD OCCUPANCIES

Our reward function should control the number and size of overfull LSH bins. We quantify this by setting a threshold, t_h, on the LSH-bin occupancy, h: LSH-band bins with occupancy below t_h are ignored, since they do not contribute to the worst-case costs. Above this threshold, we create a reward function that decreases according to $-2^{f(x)}$ where $x = h/t_h$ and $\frac{d}{dx}f(x_2) \geq \frac{d}{dx}f(x_1) \quad \forall x_2 > x_1 \geq 1$. Using a super-linear function of the above-threshold ratio as an exponent rewards splitting large bins into multiple smaller bins, even if all of the smaller bins are above threshold. This corresponds to the desired penalty behavior: even though we conceptually do not want any bin occupancy to be above threshold, it is much worse to have a single bin far above threshold than to have multiple bins somewhat above threshold, since the first case will result in much larger variance in the total candidate-list length than the second case, for any fixed look-up list.

We make this family of reward functions submodular, despite the bin-occupancy thresholding, by offsetting the reward by $2^{f(1)}$, so the reward is zero at $x = 1$. Without this offset, the reward is discontinuous at $x = 1$, destroying the submodular property. With the offset, the reward is zero for $x \leq 1$: it is zero at $x = 1$ by virtue of the offset and it is zero for $x < 1$, since these values are excluded from our evaluation.

As will be discussed in Section 6, even with our best grouping solution, we will still have “hot spots”. We address this in our final system by allowing data-adaptive LSH bin splitting to ensure our operational limits are not exceeded. For these hot-spot locations, we extend the hash key by the values from additional signature dimensions. As much as possible, we wish to avoid this solution, since it has the same uncertain recall as other data-adaptive structures [4]. Therefore, this investigation has focused on how well we can flatten the LSH lookup table occupancy *before* expanding the banding keys.

In summary, a family of submodular reward functions on the thresholded occupancy space gives the desired behavior:

$$R(H) = -\sum_{i|x_i>1}(2^{f(x_i)} - 2^{f(1)})$$

where $x_i = h_i/t_h$ is the scaled LSH occupancy for the i^{th} bin. For the results in Section 6, we used $t_h = 0.01\%$ of the training population and $f(x) = x \log_2 x$. Since the dynamic range of this reward function is very large, we implemented its evaluation in the log space.

4. EFFICIENT ABOVE-THRESHOLD HISTOGRAM REPRESENTATION AND UPDATE

One issue with using detailed LSH-bin occupancies, instead of expectancy-averaged statistics as were used in [1], is that the occupancy of the LSH tables must be represented and updated with each candidate change to how the LSH band is formed. The number of candidates that must be evaluated are as high as BL^2 on the first step (BL available dimensions crossed with L available groups) and decrease from there.

Since the distributions that we are working with are very nonuniform, we can efficiently represent the portion of the table on which our penalty is computed by listing the bin key for only those bins that are at above-threshold occupancy. The distributions using candidate extensions to the LSH-band key are then efficiently evaluated, starting from the current listing of overfull band keys and mapping the new distribution using only those signatures that fell into one of those bins, now split according to the proposed addition to the LSH-band key. Finally, this map of the subdivided problematic bins is scanned to create the new list of overfull bins (now in the extended LSH-band space). This will find all of the overfull bins in this new space, without a complete re-mapping of the full training set, since adding a dimension can only reduce bin occupancies.

For our experiments (Section 6), the number of overfull bins peaks at 1000-4000, when the LSH-band key is only two dimensions. That number drops as additional dimensions are added and the subdivisions of the overfull bins all fall below threshold. However, even with 1000-4000 overfull bins, the number of overfull bins requiring evaluation is a small and decreasing count, in both absolute and relative numbers.

As we go through the greedy process for grouping the dimensions, we keep these lists of overfull bins only for the current accepted LSH bands, discarding those for the candidate steps. This reduces the memory used during training, allowing for larger training sets. Instead of using large amounts of memory, we re-compute the overfull-bin lists for the accepted candidates as they are accepted into the solution.

5. SEEDED GREEDY SEARCH

Our reward surface has many local optima, making greedy search error prone. However, since our reward function is submodular, we know that the optimum solution is, at most, 63% better than the greedy solution. This bound can limit our search for better solutions, after evaluating the greedy solution. In addition, with our submodular reward structure, we can use CELF to efficiently find the best greedy-search completion from any “seed” from which we wish to start searching. By using combination of bounding (for termination) and efficient greedy completion of solutions, we efficiently guide our search into other parts of the space.

To get our bound, we first evaluate the fully greedy solution, starting our search without seeding. This greedy solution actually provides a comparatively low-reward solution. The intuition behind this is that the fully greedy process in effect wastes its most powerful tools — the most uniform signature dimensions — on the first L steps, by initializing the previously empty bands. This is less effective than using these dimensions slightly later in the solution, when the LSH bins that will become problems can be (slightly) better understood. For once, this low-reward on a greedy solution is the desired behavior, since it gives us a tighter limit on the maximum possible reward that can be achieved. We use that bound to stop our subsequent search for a good seeded greedy solution at 90% of the optimal-reward bound.

We then restart our greedy search using seeded bands: that is, starting from L partially filled LSH bands. To avoid an outer combinatoric search for good seeds, we only examined seeds of one dimension in each of the L bands that we had reason to believe might provide good solutions. Our choices for seeds were:

- Lowest-entropy dimensions: By picking these first, we forced the dimensions with the worst *average* clumping apart. By bringing the problem cases to the start, those problems can be explicitly targeted throughout the greedy-selection process.
- Lowest-reward dimensions: This addresses the worst of the *worst-case* clumping, instead of the worst of the average-case clumping.
- Sequential lowest-reward dimensions: With this approach, we selected the group seeds sequentially, looking at the set of incremental rewards for adding each dimension to each (already seeded or unseeded) group and selecting the next seeding dimension as the one with the lowest minimum reward. This was similar to the lowest entropy set but differed on some of the selections. The difference was due to some low-reward dimensions actually being more effective taken together than taken with other slightly-higher-reward dimensions.

We also tried some random initializations of the L bands, followed by greedy search to complete the solution. However, most random seeds led to solutions that were worse then the fully greedy solution. All random-seeded solutions were worst than those found using the above seeds.

We hoped to stop our search for seeds when a seeded greedy solution was 10% of our bound on the optimum (given by the fully greedy solution). As mentioned in Section 6, we instead stopped after having run through all of our seeds.

6. RESULTS

We examined two data sets of 150,000 signatures, taken from 3000 distinct media tracks sampled once per second. The first population is used to find the LSH bands. The second, independent set, is used only for testing. The 100-dimensional signatures were generated using an approach similar to what is described in [2]. The two worst- and best-distributed individual signature dimensions are shown in Figure 1. We compare the CELF approach with banding

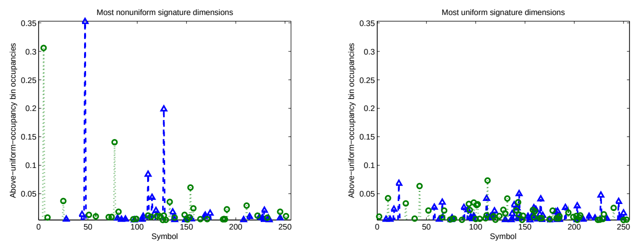

Fig. 1. The two most non-uniformly distributed dimensions in the 100-dimensional signature (left), as well as the two most uniformly distributed ones (right).

by randomly selecting groups of 4 dimensions and with banding for minimum within-band mutual information (*min.-mutual banding*), as proposed by [1].

To do the CELF modeling, we set the threshold on the CELF approach to 0.01% of the training set. This is below our required operational limit, in order to provide better robustness to differences between the training distribution and the operational distribution. We used the seeded greedy search, as described in Section 5. Seed selection using sequential lowest reward gave the best solution, which was about 20% better than the fully-greedy solution. That improvement did not put the seeded greedy solution within our hoped-for 90% limit (relative to the fully-greedy bound on the optimal solution). This seems surprising, given the large dynamic range of our reward function. However, there were 4 bins that contributed the vast majority of the penalty on our greedy solutions. These worst-bins remained at occupancy levels of 0.12-0.2% of the full population for all of our seeded solutions. All bins would need to be reduced to below 0.12% occupancy to achieve our target of 90% of the optimal bound. The sequential lowest reward got closest to that by containing these occupancies to the range of 0.12-0.16%.

The results from the sequential-lowest-reward seed, applied to the test set, are shown in Figure 2 and 3. Figure 2 shows the entropy of the test population across the LSH bands (sorted from greatest to least). The average entropy across all the bands is almost unchanged by the different approaches. However, due to variation in the band entropy, the average candidate list length for the random banding will be about 4% longer than the min.-mutual banding. The average candidate list length for CELF banding is also worse than for the min.-mutual banding: the average length will be about 1% longer. This larger average length is expected, since the min.-mutual method was designed to address the average case.

Figure 3 shows the worst-case occupancy of the test population in the largest LSH-band bins. This plot shows the improvement that is achieved using the explicit attention to these worst-case occupancies. The number of bins that were at an occupancy level that creates memory problems went down from 0.17% of the occupied LSH bins for random banding, to 0.12% using min.-mutual banding, to 0.04% using CELF banding. More importantly, the worst-case occupancy level went down from 0.25% for both random and min.-mutual banding to 0.16% for CELF banding. Since it is this maximum occupancy at the per-bin level that defines what our worst-case memory usage will be, this reduction in maximum retrieval length is critical. Most importantly, the number of entries that were in these worst-case bins was reduced from 12% for random banding, to 8% using min.-mutual banding, to 2.5% using CELF banding. Since the distribution of the expected lookups into the LSH tables is the same as the distribution of the entries, this reduction in the percentage of lookups

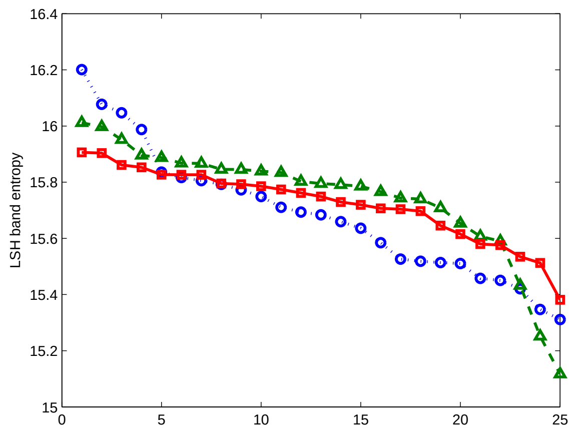

Fig. 2. Distribution entropy for the bands created by random association (blue dotted circles), min. mutual (green dashed triangles), and CELF (red solid squares). Minimum mutual information does best by this measure since it optimizes the average case, which is what entropy measures.

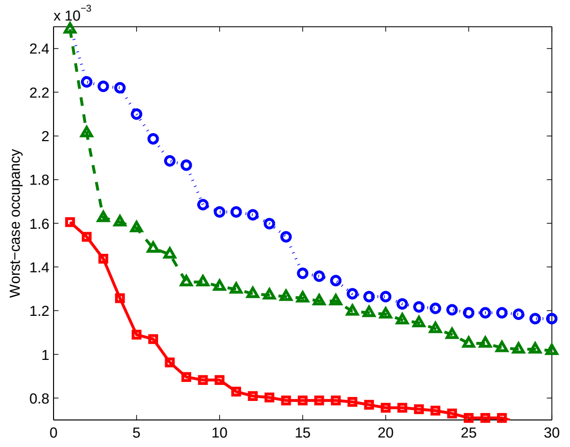

Fig. 3. Occupancy of largest bins for the bands created by random association (blue dotted circles), min. mutual (green dashed triangles), and CELF (red solid squares). CELF does best since it minimizes the worst cases. This is what we must design for operationally.

that will see the worst-case memory usage is also critical.

Even with this CELF solution for minimizing the worst case occupancy, we continue to see a subset of bin occupancies above the level that can be easily supported in a large-scale system. For these problem cases, we combine the techniques used in data-adaptive structures, like spill trees [4], with LSH banding. When creating the LSH table, if a bin exceeds a threshold set by our operational constraints, then the bin is marked as "expanded" and the previous entries are re-indexed into bins corresponding to the current LSH band plus an additional signature dimension. Which dimensions are used to expand each LSH band is selected off line, again based on training data, but the actual subdivision of any given bin is done based on operational conditions. When a lookup encounters an expanded bin, it will retrieve the reduced set of results from the subdivided bin. This approach has the same disadvantage of data-adaptive structures, that the recall radius is not known before the table is built. However, when we used this approach in combination with the CELF-selected bands, we saw no change in our actual recall rates on our regression tests, most likely due to the small percentage of lookups affected by the the adaptive-bin splitting (only 2.5%, due to our band design).

7. CONCLUSIONS

We have presented an approach to designing LSH bands, based on a given population of signature dimensions. The approach explicitly targets the worst case retrievals that result in too-long candidate lists being retrieved from these tables. This worst-case set is what must be examined to design a practical large-scale audio or video retrieval system, where the output from the LSH tables themselves represent a significant consumption of resources.

Using a submodular reward function to characterize the above-threshold bin occupancy provides two significant advantages. First, it provides an upper bound on our best possible solution, thereby giving an indication of how much *might be* gained from a more complete search. Second, it allows us to find the greedy solution efficiently, freeing the training-time resources that would have otherwise been used in finding the first solution to evaluate alternatives.

A direction for future work is to evaluate the effect of changing the reward function. Another is to expand the set of starting seeds that are evaluated by our greedy search. A third direction would be to work towards an occupancy representation that allowed efficient evaluation for simulated annealing or other stochastic optimization approaches to search: with such a representation we would be less likely to fail due to a poor seed or to the greedy search trapping us in a local maxima.

8. REFERENCES

[1] S. Baluja, M. Covell, S. Ioffe, "Permutation grouping: Intelligent hash-function design for audio and image retrieval," in *Proc. ICASSP*, 2008.

[2] S. Baluja, M. Covell, "Waveprint: Efficient wavelet-based audio fingerprinting," *Pattern Recognition*, 2008.

[3] A. Torralba, R. Fergus, W. T. Freeman, "80 million tiny images: A large dataset for non-parametric object and scene recognition," *Trans. PAMI*, 2008.

[4] T. Liu, et al., "An investigation of practical approximate nearest neighbor algorithms," in *Proc. NIPS*, 2004.

[5] P. Indyk, R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proc. ACM Symposium Theory of Computing*, 1998, pp. 604–613.

[6] S. Baluja, M. Covell, "Learning to hash: forgiving hash functions and applications," *Data Mining and Knowledge Discovery*, 2008.

[7] M. Sviridenko, "A note on maximizing a submodular set function subject to a knapsack constraint," *Operations Research Letters*, 2004.

[8] J. Leskovec, et al., "Cost-effective outbreak detection in networks," in *Proc. ACM International Conference on Knowledge Discovery and Data Mining*, 2007.