Dimension-specific Search for Multimedia Retrieval

Zi Huang[1], Heng Tao Shen[1], Dawei Song[2], Xue Li[1], Stefan Rueger[3]

[1] School of ITEE, The University of Queensland, Australia
[2] School of Computing, The Robert Gordon University, UK
[3] Knowledge Media institute, The Open University, UK

Abstract. Observing that current Global Similarity Measures (GSM) which average the effect of few significant differences on all dimensions may cause possible performance limitation, we propose the first Dimension-specific Similarity Measure (DSM) to take local dimension-specific constraints into consideration. The rationale for DSM is that significant differences on some individual dimensions may lead to different semantics. An efficient search algorithm is proposed to achieve fast Dimension-specific KNN (DKNN) retrieval. Experiment results show that our methods outperform traditional methods by large gaps.

1 Introduction

Given an image feature database, the similarity measure determines the retrieval effectiveness. All existing similarity measures compute the *global similarity* which is aggregated from all dimensions of the feature space without exploring the impact of local distance along each individual dimension. We refer them as Global Similarity Measures (GSM) [7]. Although the distances along some dimensions are significant, such differences become non-discriminative in global similarity computation which averages the differences to all dimensions. Intuitively, properly utilizing the distance along each individual dimension might help to improve retrieval by filtering more irrelevant objects from top-K results.

On the other hand, high dimensionality of the feature space drives existing indexing structures [2, 3, 5, 6] deficient due to the 'curse of dimensionality'. One important reason is that they maintain all dimensions as a whole for global similarity computations and all dimensions need to be accessed during query processing. [4] approaches high-dimensional indexing as a physical database design problem and proposes to vertically decompose the data by maintaining a separate table for each dimension (vertical decomposition for short). Some data points can be pruned away from full distance computations by accessing fewer dimensions/tables based on global upper and lower bounds estimated from all dimensions.

In this paper, we propose the first Dimension-specific. Different from conventional GSM, DSM takes dimension-specific constraints into consideration, where two feature vectors must match within a certain tolerance threshold along each individual dimension. In other words, DSM computes the global similarity as in conventional GSM, subject to the maximum allowable variation in each dimension. The rationale for DSM is that significant differences on individual dimensions may often lead to different human perception (i.e., semantics).

Corresponding to DSM, we introduce a new type of query called Dimension-specific KNN (DKNN) query to find KNN with dimension-specific constraints. To

achieve fast retrieval for DKNN query, we propose an search algorithm to coordinate efficient dimension-specific pruning and global KNN pruning concurrently based on derived pruning rules. It accesses the data in a dimension-by-dimension manner and the intermediate candidate set obtained on the current dimension is propagated to the next dimension for further lookup and continuous reduction. It avoids most of high-dimensional distance computations and is robust to increasing dimensionality.

An extensive empirical performance study is conducted on the widely used Getty image datasets. The experimental results showed that DSM generated better MAP over GSM based on classical Euclidean distance by very large gaps. Furthermore, our DKNN search algorithm achieves great pruning power and outperforms traditional KNN method extended for DKNN query by an order of magnitude.

2 Dimension-specific KNN Search

2.1 Dimension-specific Similarity Measure (DSM)

Definition 1 (Dimension-specific Similarity Measure). *Given a query object represented by its feature vector* $x_q = (x_q^1, x_q^2, ..., x_q^D)$ *and a database object represented by its feature vector* $x=(x^1, x^2, ..., x^D)$*, their dimension-specific dissimilarity* $DS(x_q, x)$ *is defined as:*

$$DS(x_q, x) = \begin{cases} d(x_q, x), & if\ \forall i \in \{1..D\}, x_q^i \cong_{\varepsilon^i} x^i \\ +\infty, & otherwise \end{cases}$$

where $d(x_q, x)$ *is the global dissimilarity measure applied if the dimension-specific constraints (i.e.,* $\forall i \in [1..D], x_q^i \cong_{\varepsilon^i} x^i$*) are satisfied, and* D *is the dimensionality of the feature space.* [4]

DSM is able to avoid individual significant differences being potentially neglected in global measures. Given a query, the problem we investigate here is to find the *top-K most similar results* from the image database. By applying DSM, the similarity between two images is measured by the Euclidean distance as its *global similarity*, subject to the *dimension-specific conditions*. We define this type of query as Dimension-specific KNN (DKNN) query.

In DSM, ε is an important parameter which determines the qualification of a data point to compute its global distance to a query. To assign the value of ε, one simple way is to analyze historical query results for a proper fixed ε value to all dimensions. This may work for the feature space whose data along all dimension are uniformly distritbuted in the same range. When different dimensions exhibit significantly different distributions, adaptive ε values in DSM could be more effective. For i^{th} dimension, we can associate its ε^i with σ^i(standard deviation) by setting $\varepsilon^i = c \times \sigma^i$, where c is a scalar parameter.

2.2 Pruning Rules for DKNN Search

From Definition 1, we can simply set up the following **conditional pruning rule**: *x can be safely pruned if* $\exists\ i \in \{1..D\}$ *such that* $|x_q^i - x^i| > \varepsilon^i$.

[4] $x_q^i \cong_{\varepsilon^i} x^i$ if $|x_q^i - x^i| \leq \varepsilon^i$.

Next we derive the KNN pruning rule. As we mentioned earlier, our data is organized based on vertical decomposition.Our DKNN search algorithm will access the data in a dimension-by-dimension manner. During the query processing, assume we have accessed the first t dimensions. We first derive the upper/lower bound of the distance between two points.

Proposition 1. *The upper bound of the distance between a query x_q and a data point x is defined by:*

$$||x_q, x||^2 \leq \underbrace{\sum_{i=1}^{t}(x_q^i - x^i)^2}_{\text{distance on t dimensions}} + \underbrace{\sum_{i=t+1}^{D}(\max(|x_q^i - r_{\min}^i|, |r_{\max}^i - x_q^i|))^2}_{\text{upper bound on remaining dimensions}}$$

where $r_{\min}^i$ and $r_{\max}^i$ are the minimal and maximal values on the i^{th} dimension in the feature space respectively.

The lower bound of the distance between a query x_q and a data point x is defined by:

$$||x_q, x||^2 \geq \underbrace{\sum_{i=1}^{t}(x_q^i - x^i)^2}_{\text{distance on t dimensions}} + \underbrace{\frac{(\sum_{i=t+1}^{D} x_q^i - \sum_{i=t+1}^{D} x^i)^2}{D-t}}_{\text{Lower bound on remaining dimensions}}$$

Based on the derived upper and lower bounds, we can derive the following **KNN pruning rule**: *x can be safely pruned if its lower bound is greater than the current K^{th} smallest upper bound.*

2.3 DKNN Search Algorithm

DKNN Search Algorithm
Input: x_q, Rank[ ]
Output: DKNN[ ]

1. For i=1 to D
2. DKNN[ ]←UpdateCandidateBound($x_q^{Rank[i]}$, $x^{Rank[i]}$);
3. DKNN[ ]←SortCandidate();
4. For j=1 to DKNN.size()
5. if |DKNN[j][i]-x_q^i|>ε^i
6. Remove DKNN[j−−];
7. else if j>K and DKNN[j].lb>DKNN[K].ub
8. Remove DKNN[j−−];
9. Return DKNN[ ];

Fig. 1. DKNN Query Processing

Based on the established pruning rules, we propose a new search algorithm (Fig 1) which can maintain a small intermediate candidate set by cooperating conditional

pruning and KNN pruning concurrently. As more dimensions have been accessed, the candidate set size becomes smaller and smaller. Therefore, I/O cost can be reduced significantly.

The order in which we access the dimensions does not change the final results. However, accessing the dimensions with higher pruning power (PP) [5] earlier will reduce the size of intermediate results, resulting in smaller computational cost and I/O cost. Normally, a larger σ^i corresponds to a larger PP on that dimension. So it is recommended to access the dimensions in the descending order of their σ^i to improve the performance.

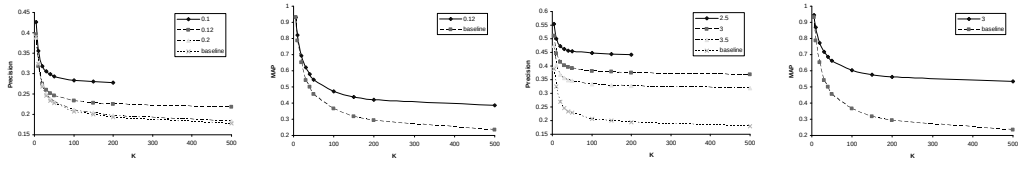
3 Experiments

An extensive empirical performance study is conducted on Getty Image Dataset. The experimental results confirm the effectiveness of DSM and the efficiency of our DKNN search algorithm.

3.1 Experiment Setup

Getty Image Dataset contains 21,820 images[6], together with their annotations. RGB feature in 216 dimensionality is generated for each image. The average standard deviation on all dimensions is about 0.025. Precision and Mean Average Precision (MAP) are used to measure the effectiveness of DSM. Two images are considered as relevant if they share one or more keywords. PP is used to measure the efficiency of our DKNN search algorithm.

3.2 Effectiveness and Efficiency



(a) precision, fixed ε (b) MAP, fixed ε (c) precision, adaptive ε (d) MAP, adaptive ε

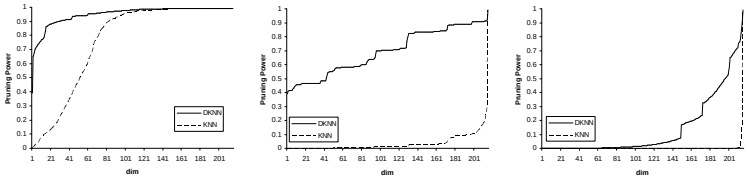
Fig. 2. Effect of fixed/adaptive ε on precision/MAP at top-K results on Getty dataset.

Effectiveness: Fig 2 shows the results on precision and MAP at top-K results when we use fixed/adaptive ε for all dimensions. We can observe that DSM outperforms baseline for different values of ε. However, a too small value of the fixed ε may return insufficient number of results, while a too large value may lose the effect dimension-specific constraints. A suitable value will improve the search quality by

[5] $PP = No.\ of\ pruned\ objects/No.\ of\ total\ objects.$

[6] http://creative.gettyimages.com

huge gaps. Compared with MAP result in fixed ε shown in Fig 2(b), MAP is further improved by setting an adaptive ε, which is less sensitive to large variances and able to achieve better results quality than fixed ε.



(a) descending (b) natural (c) ascending

Fig. 3. Effect of Dimension Access Order on Pruning Power with fixed ε=0.12 and K=300.

Efficiency: We compare three dimension access orders (i.e.,access by the ascending/descending/natural (original)) of dimensions standard deviations. As shown in Fig 3, accessing the dimensions in the descending order of dimension standard deviations outperforms natural order greatly, which in turn outperforms accessing the dimensions in the ascending order of dimension standard deviations significantly. Furthermore, DKNN algorithm outperforms DKNN outperforms traditional KNN by an order of magnitude.

4 Conclusion

In this paper, we introduce a new type of query called Dimension-specific KNN (DKNN) query to find KNN with dimension-specific constraints. An efficient DKNN search algorithm is developed based on dimension-specific pruning and global KNN pruning rules concurrently. An extensive empirical performance study reveals that our proposals achieves significant improvements over traditional methods.

References

1. I. Assent, A. Wenning, and T. Seidl. Approximation techniques for indexing the earth mover's distance in multimedia databases. In *ICDE*, page 11, 2006.
2. C. Böhm, S. Berchtold, and D. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Computing Surveys*, 33(3):322–373, 2001.
3. P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *VLDB*, pages 426–435, 1997.
4. A. de Vries, N. Mamoulis, N. Nes, and M. Kersten. Efficient k-NN search on vertically decomposed data. In *SIGMOD*, pages 322–333, 2002.
5. H. Jagadish, B. Ooi, K. Tan, C. Yu, and R. Zhang. idistance: An adaptive b+-tree based indexing method for nearest neighbor searc. *TODS*, 30(2):364–379, 2005.
6. Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-Probe LSH: Efficient Indexing for High-Dimensional Similarity Search. In *VLDB*, pages 950–961, 2007.
7. R. Weber, H. Schek, and S. Blott. A quantitative analysis and performance study for similarity search methods in high dimensional spaces. In *VLDB*, pages 194–205, 1998.