Learning for Efficient Retrieval of Structured Data with Noisy Queries

Charles Parker, Alan Fern, Prasad Tadepalli
Oregon State University
Structured Data Retrieval: The Problem

- Vast collections of structured data (images, music, video)
- Development of noise-tolerant retrieval tools
- Query-by-content
- Accurate as well as efficient retrieval
Sequence Alignment: Introduction

- Given a query sequence and a set of targets, choose the best matching (correct) target for the given query.

- Useful in many applications
  - Protein secondary structure prediction
  - Speech recognition
  - Plant identification (Sinha, et. al.)
  - Query-by-humming
Obligatory Overview Slide

- Sequence Alignment Basics
- Metric Access Methods
- The Triangular Inequality
- VP-Trees
- Boosting for Efficiency
- Results and Conclusion
Sequence Alignment: Basics

- Having a query sequence and a target sequence we can *align* the two sequences.
- Matching (or close) characters should align – characters only present in one or the other should not.
- Suppose we have query = “DRAIN” and target = “CABIN” . . .

```
i  i  d  m  d  m  m
-  -  C  A  B  I  N
D R  -  A  -  I  N
```
Sequence Alignment:
Alignment Costs

- Scoring the alignment for evaluation
- Suppose we have a function $c$ that gives us costs for edit operations:
  - $c(a, b) = 3$ if $a = b$ and 0 if other non-null character
  - $c(-, b) = 1$
  - $c(a, -) = 1$
- The alignment below has a cost of 13

```
i i d m d m m
- - C A B I N
D R - A - I N
```
The Dynamic Time Warping (Smith-Waterman) algorithm

- Find best reward path from any point in target and query
- Fill the values in the matrix using the following equations starting from (0,0)

\[
\begin{align*}
\text{align}(i, j, t, q) &= \max \left\{ c(\cdot, q_j) + \text{align}(i+1, j, t, q) \right. \\
&\quad \left. c(t_i, \cdot) + \text{align}(i, j+1, t, q) \right. \\
&\quad \left. c(t_i, q_j) + \text{align}(i+1, j+1, t, q) \right\}
\end{align*}
\]

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>A</th>
<th>B</th>
<th>I</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>R</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>I</td>
<td>4</td>
<td>7</td>
<td>7</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>N</td>
<td>5</td>
<td>8</td>
<td>9</td>
<td>11</td>
<td>13</td>
</tr>
</tbody>
</table>
Gradient Boosting: Learning Distance Functions

- Define the margin to be the score of the correct target minus the score of the highest scoring incorrect target.
- Formulate a loss function according to this definition of margin.
- Take the gradient of this function at each possible “replacement” that occurs in the training data.
- Iteratively move in this direction.
Metric Access Methods: Overview

- Accuracy is not enough!
- Avoidance of linear search
- Ability to cull subsets with the computation of a single distance.
- Use of the triangular inequality
- Use of previous work to assure some level of satisfaction
The Triangular Inequality
(Skopal, 2006)

- Need to increase small distances while large ones stay the same
- Applying a concave function does the job

$$d^*(x, y) = d(x, y)^{1+w}$$

- Function moves distances within the same range
- Could create a useless metric space
- Use line search to assure optimality
The Triangular Inequality: Concave Function Application
Vantage Point Trees: Overview

- Given a set $S$
  - Select a “vantage point” $v$ in $S$.
  - Split $s$ into $s_l$ and $s_r$ according to distance from $v$.
  - Call recursively on $s_l$ and $s_r$.
- Builds balanced binary tree.
Vantage Point Trees: Demonstration
Vantage Point Trees: Demonstration
Vantage Point Trees: Demonstration
Vantage Point Trees: Demonstration
Vantage Point Trees: Demonstration
Vantage Point Trees: Demonstration
Vantage Point Trees: Demonstration
Vantage Point Trees:
Searching for Nearest Neighbors Within ‘t’
Vantage Point Trees:
Searching for Nearest Neighbors Within ‘t’
Vantage Point Trees: Searching for Nearest Neighbors Within ‘t’

Assume $S_i$ contains a sequence within distance $t$ of $Q$, and that the triangular inequality holds.
Vantage Point Trees: Searching for Nearest Neighbors Within ‘t’

Assume $S_i$ contains a sequence within distance $t$ of $Q$, and that the triangular inequality holds.
Vantage Point Trees:
Searching for Nearest Neighbors Within ‘t’

Assume $S_i$ contains a sequence within distance $t$ of $Q$, and that the triangular inequality holds
Vantage Point Trees: Searching for Nearest Neighbors Within ‘t’

Assume $S_i$ contains a sequence within distance $t$ of $Q$, and that the triangular inequality holds.
Vantage-Point Trees: Searching for Nearest Neighbors Within ‘t’

\[ d(Q, A) > m + t \]

\[ d(Q, S_i) < t \]

\[ d(A, S_i) < m \]

\[ d(Q, A) > d(A, S_i) + d(Q, S_i) \]
Vantage-Point Trees: Searching for Nearest Neighbors Within ‘t’

d(Q, A) > m + t

d(Q, S_i) < t

d(A, S_i) < m

\[ d(Q, A) > d(A, S_i) + d(Q, S_i) \]

but according to the triangular inequality, \[ d(Q, A) \leq d(A, S_i) + d(Q, S_i) \]
Vantage Point Trees:
Searching for Nearest Neighbors Within ‘t’

Thus we have a contradiction and can eliminate $S_1$ from consideration.
Vantage Point Trees: Optimizing

- Similar proof for $d(Q, A) < m - t$
- However, if $m - t < d(Q, A) < m + t$, we can do nothing, and must search linearly
- If there are no nearest neighbors within $t$, no guarantees
- $t$ should be as small as possible . . .
- . . . or, target/query distances should be as far as possible to the correct side of the median distance
Boosting for Efficiency: Summary

- Create an instance of a metric access data structure given a target set
- Define a loss function particular to that structure
- Take the gradient of this loss function
- Use the gradient to tune the distance metric to the structure
Boosting for Efficiency: vp-tree-based Loss Function

- Sum loss for each training query and each target along path to correct target
- Scale loss according to “cost” of mistake
- \( i \) is training query, \( j \) is target along path
- \( v_{ij} \) is “left or right”, \( m_{ij} \) is median
- \( f_{ij}(a,b) \) is replacement count for \( j \)th target in the \( i \)th training query’s path

\[
\sum_{i} \sum_{j} \log(1 + \exp(v_{ij}[m_{ij} - \sum_{a} \sum_{b} c(a,b)f_{ij}(a,b)]) \frac{2^{j}}{2^{j}})
\]
Boosting for Efficiency: Gradient Expression

\[
\frac{\partial L}{\partial c_k(a,b)} = -\sum_i \sum_j \frac{v_{ij} f_{ij}(a,b)}{2^j (1 + \exp(-v_{ij} (m_{ij} - d(t_{ij}, q_{ij}))))}
\]

- Retains properties of the accuracy gradient (easy computation, ability to approximate)
- Expresses desired notion of loss and margin
Synthetic Domain:  
Summary  
- Targets are sequences of five tuples \((x, y)\) with domains \((0,9)\) and \((0,29)\) respectively  
- Queries generated by moving sequentially through  
  - With \(p = 0.3\), generate a random query event with \(y >=15\) (insert)  
  - Else, if target \(y\) is < 15, generate match. If target \(y\) is >= 15, skip to next target element (delete).  
  - Matches are \((x_1, y_1) \rightarrow (x_1, (y_1 + 1) \mod 30)\)  
- Domain is engineered to have structure
Query-by-Humming: Summary

- Database of songs (targets)
- Can be queried aurally

Applications
- Commercial
- Entertainment
- Legal

Enables music to be queried on it’s own terms
Query-by-Humming: Basic techniques for query processing

- Queries require preprocessing
  - Filtering
  - Pitch detection
  - Note segmentation
- Targets, too!
  - Polyphonic transcription
  - Melody spotting
- We end up with a sequence of tuples (pitch, time)
Application to Query-by-Humming: Our Data Set

- 587 Queries
- 50 Subjects (college and church choirs)
- 12 Query songs
- Queries split for training and test
Results:

Experimental Setup

- First 30 iterations: Accuracy boosting only
- Construct VP-Tree
- Compare two methods for second 30 iterations
  - Accuracy only
  - Accuracy + efficiency (May require rebuild of tree)
- Efficiency is measured by plotting % of target set culled vs. error
  - Vary t
  - Would like low error and high % culled
  - In reality, lowering t introduces error
Results:
Query-by-Humming Domain

![Graphs showing performance of Efficiency Boosting and Accuracy Boosting over iterations and error tolerance.](image)
Results:
Synthetic Domain
Conclusion

- Designed a way to specialize a metric to a metric data structure
- Showed empirically that accuracy is not enough
- Showed successful “twisting” of the metric space
Future Work

- Extending to other types of structured data
- Extending to other metric access methods
  - Some are better (metric trees)
  - Some are worse (cover trees)
- Use as general solution to structured prediction problem
- Use in automated planning and reinforcement learning
Fin.