Nearest Neighbour Search with Zero-Suppressed Decision Diagram for Text Retrieval

Yong Boon, Lim
Supervisor: Scott Sanner
Outline

• Motivation
• Nearest Neighbour Search (NNS)
• Zero-Suppressed Decision Diagram (ZDD).
• NNS with ZDD Multi-Hash
• Results
• Conclusion
• Q & A
Motivation

• Application for NNS on Text
• Text Documents have **high** dimensionality.
• Exact NNS on text is **expensive**.
• Current Trend : Approximate NNS on text (LSH).
• Propose:
  – Exact NNS on High Dimensionality (Using ZDD).
  – Approximate NNS with ZDD & Multi-hash.
NNS & Exhaustive NN

• What’s NNS?
  – Find the closest point \((p_i)\) in a database to a query \((q)\).

• Exhaustive NN
  – Calculate the distance of every point \((p_i)\) in the database, and keep track for the closest point.
  – Query cost increases exponentially with database size, and dimensionality of the point \((p_i)\).
Zero-Suppressed Decision Diagram

- Compact way of representing sets of subsets.
- **Elements:**
  
  - Support Set Operations: Union, Intersec, Diff
  - Each path from root node to Terminal Node 1 represents a subset.
  - A variable is present in the subset if the path goes through the 1-edge of the node represents it.

- **Document Representation:**
  
  Path-to-T1 = Document Variable = Doc. Term

- Example:
  
  \{\{Adam, Love, Eve\}, \{Adam, Love\}, \{Eve, Love\}\}
NN-ZDD Build Phase

Observation 1: Same # of Nodes

Observation 2: Only 2 extra Nodes.

Property: Node Reusability → Compact Representation
NN-ZDD Query Phase

• Given a query = {Apple, Love}
  Which is the closest doc? (NNS)
• Equivalent Question:
  Which path-to-T1 contains the most query variables in its 1-Edges?
• How do find this path-to-T1?
• Solution : Dynamics Programming
  – Find the edge with max. score recursively from root node to T1 node.
  – If the node variable match the query variable, give score to 1-edge.
  – Base Score :
    T1 Node = 1, T0 Node = -2, Match Score = 1
• You got NN ZDD!
Multi-Hash

• Approximate NNS.
• Assumption:
  High similarity docs \textit{stay close} to each other after hashing.
• Idea:
  Distribute the Big NN to tree with smaller NN at the branch through hashing.

Build
  – Hashing docs to lower dimensionality.
  – Increase the hash size on each layer to separate less similar from more similar.
  – A set of File IDs stored at the last layer. All docs with the same hash value stay together.

Query
  – Perform NN-ZDD on layer-by-layer to retrieve closest hash key, and eventually closest Doc.
• Hash function plays \textit{important} role to balance the tree thus efficiency.
NN-ZDD Multi-Hash Build Phase

- Each doc produces the hash value at each layer and form a hash chain.
- Store the doc id at last layer.
NN-ZDD Multi-Hash Build Phase

- The query goes through the same hash function.
- Perform NN-ZDD at each layer with query hash on the child hash hash bucket selected on the parent hash bucket.
- Perform exhaustive NN search with the Doc Id at last layer.
Evaluation

• Performance Metrics

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build Time (ms)</td>
<td>Time to build the data structure</td>
</tr>
<tr>
<td>Query Time (ms)</td>
<td>Time to retrieve the NN docs</td>
</tr>
<tr>
<td>Accuracy (%)</td>
<td>The % of the docs correctly matched.</td>
</tr>
<tr>
<td>Node Size</td>
<td># of Nodes in ZDD (ZDD only)</td>
</tr>
</tbody>
</table>
ZDD Nodes Sizes vs Doc. Sizes

- ZDD node sizes increase linearly with the number of document.
Exhaustive-NN vs NN-ZDD

- Comparing the Query Time for Exhaustive-NN & NN-ZDD.
- NN-ZDD is much faster than exhaustive-NN.

*Doc Size : 701
Exact NNS for Different Metrics

- Comparing NN-ZDD: Hamming Distance, IDF, and TF-IDF
- NN-ZDD performs better for Binary than Frequency-based score.

*Doc Size: 799
Approximate NNS for Different Metrics

- Comparing NN-ZDD, LSH, Multi-Hash, ZDD Multi-Hash.
- LSH, Multi-Hash, and ZDD Multi-Hash have 100% accuracy.
- LSH - less query time.
- Multi-Hash & ZDD Multi-Hash – less build time.

*Doc Size : 799
Approximate NNS for Larger Dataset

- Comparing LSH, Multi-Hash, ZDD Multi-Hash.
- ZDD Multi-Hash has higher accuracy, comparable query time, and less build time with LSH.

*Doc Size : 2000
Conclusions

Exact NN Search

• NN-ZDD are order of magnitude faster than exhaustive NN.

Approximate NN Search

• NN-ZDD MLH query time closer to LSH with less build time and better accuracy.