Similarity Search: a Web Perspective

Yury Lifshits
Caltech
http://yury.name

Google Tech Talk
18 October 2007

---

Similarity Search in a Nutshell

Input: Set of objects
Task: Preprocess it

Query: New object
Task: Find the most similar one in the dataset

---

Roadmap

1. Similarity Search in Web
2. Similarity Search in Theory
3. Revising the Problem
4. New Algorithms

---

Similarity Search in Web

1
Similarity Search vs. Web

- Recommendations (movies, books...)
- Personalized news aggregation
- Ad targeting
- “Best match” search: Resume, job, BF/GF, car, apartment
- Co-occurrence similarity: Suggesting new search terms

2

Similarity Search in Theory

Similarity in Networks

Similarity chart for paper recommendation:

Simialrity is high when:
# of chains is high, chains are short, chains are heavy

Nearest Neighbor Search

Search space: object domain \( U \), distance function \( d \)
Input: database \( S = \{p_1, \ldots, p_n\} \subseteq U \)
Query: \( q \in U \)
Task: find \( \text{argmin}_{p_i} d(p_i, q) \)

Data Models:
- General metric space: triangle inequality + oracle access
- \( k \)-dimensional Euclidean space with Euclidean, Manhattan, \( L_p \) or angle metric
- Strings with Hamming or Levenshtein distance
- Finite sets with Jaccard metric \( d(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|} \)
Which One to Use?

- Sphere Rectangle Tree
- Orchard’s Algorithm
- k-d-B tree
- Geometric near-neighbor access tree
- Excluded middle vantage point forest
- MVP-tree
- Fixed-height
- fixed-queries tree
- AESA
- Vantage-point tree
- LAESA
- R*-tree
- Burkhard-Keller tree
- BBD tree
- Navigating Nets
- Voronoi tree
- Balanced aspect ratio tree
- Metric tree
- vp*-tree
- M-tree
- Locality-Sensitive Hashing
- SS-tree
- R-tree
- Spatial approximation tree
- Multi-vantage point tree
- Bisector tree
- mb-tree
- Cover tree
- Hybrid tree
- Generalized hyperplane tree
- Slim tree
- Spill Tree
- Fixed queries tree
- X-tree
- k-d tree
- Balltree
- Quadtree
- Octree
- Post-office tree

Four Famous Techniques

- Branch and bound
- Greedy walks
- Epsilon nets

Mappings: LSH, random projections, minhashing

Works for small intrinsic dimension

Revision: Data Model

- Several types of nodes and (weighted) edges, restrictions on degrees
- **Similarity chart**: List of “contributing chains”
- Similarity (relevance): sum of weight products over all contributing chains
Similarity Search in Bipartite Graphs

- Dataset: bipartite graph
  - Person-person similarity: # of 2-step chains
  - Person-movie similarity: # of 3-step chains

- Query: new person $q$ (out degree $\leq k$)
- Task: find person (movie) with maximal number of 2-step (3-step) chains to $q$

Open problem:
Existence of similarity search with $\text{poly}(m, n)$ preprocessing and $\text{poly}(k, \log n, \log m)$ query time

Revision: Basic Assumptions

In theory:
- Triangle inequality
- Doubling dimension is $o(\log n)$

Typical web dataset has separation effect
For almost all $i, j$: $1/2 \leq d(p_i, p_j) \leq 1$

Example: Jackard metric for # of joint friends

Corollaries:
- In general metric space exact problem is intractable
- Branch and bound algorithms visit every object
- Doubling dimension is at least $\log n/2$

Revision: Notion of Success

In theory:
- $c$-approximate algorithm returns $p$: $d(p, q) \leq c \cdot d(p_{NN}, q)$
- Polynomial preprocessing & sublinear search algorithm [AI06]

With separation effect:
- Returning random object has approximation factor 2
- But returning random object is in fact very poor algorithm

Suggestion
Focus on $c$-approximation of similarity

Open problem:
Existence of polynomial preprocessing & sublinear search approximate algorithm for Euclidian space with cosine similarity

Revision: Dynamic Aspects

In theory:
Handling insertions & deletions

Web:
- Adding & removing edges
  - Affects many pairwise similarities
- Weights are changing
  - Example: # of votes/comments on Digg.com
- General formula for similarity is changing
New Algorithms for Similarity Search

Concept of Disorder

Sort all objects by their similarity to $p$:

Then by similarity to $r$:

Dataset has **disorder** $D$ if

$$\forall p, r, s : \text{rank}_r(s) \leq D(\text{rank}_p(r) + \text{rank}_p(s))$$

Ranwalk Algorithm [GLS08]

Similarity search with roughly $\mathcal{O}(Dn \log n)$ data structure and $\mathcal{O}(D \log n)$ search time

Ranwalk: Data structure

Set $D' = 6D \log \log n$

For every object $p$ in database $S$ choose at random:

- $D'$ pointers to objects in $S = B(p, n)$
- $D'$ pointers to objects in $B(p, \frac{n}{2})$
- $\ldots$
- $D'$ pointers to objects in $B(p, D')$
Ranwalk: Search via Greedy Walk

- Start at random point \( p_0 \)
- Check endpoints of 1st level pointers, move to the best one \( p_1 \)
- \( \ldots \)
- Check all \( D \) endpoints of bottom-level pointers and return the best one \( p_{\log n} \)

Zipf Model

- Terms \( t_1, \ldots, t_m \)
- To generate a document we take every \( t_i \) with probability \( \frac{1}{i} \)
- Database is \( n \) independently chosen documents
- Similarity between documents is defined as the number of common terms

Magic Level Theorem [HLN07]

For **magic level** \( q = \sqrt{2 \log_e n} \):

- **Any match**: W.h.p. the best document in database has \( q \pm \epsilon \) overlap with query document

- **Prefix match**: W.h.p. there is a document in database containing \( q \pm \epsilon \) of top frequent terms of query document

Best prefix match is much easier to search for!

Questions to Google

- **Google problems**: What are the main challenges in implementing similarity search?
- **Announce the winner**: Which similarity search algorithms do you use?
- **Google datasets**: Give us benchmarks in ad targeting, news aggregation, citation networks
Thanks for your attention! Questions?