Efficient Neighborhood Graph Construction for Sparse High Dimensional Data

David C. Anastasiu
Department of Computer Engineering
San José State University

In collaboration with:
George Karypis, University of Minnesota

Near Duplicate Detection

SpaceX rocket fails to land on barge

Company never expected to nail this landing, says SpaceX chief Elon Musk

The Associated Press Posted: Mar 04, 2016 3:00 PM ET | Last Updated: Mar 04, 2016 8:46 PM ET

SpaceX has already succeeded in landing a Falcon rocket at an on-shore site near the Cape Canaveral pad where it launched, but it has failed in previous attempts to guide rockets back to ocean platforms. (SpaceX)

Related Stories

- SpaceX rocket launches satellite but botches ocean landing
- Why competition is good for the space race: Bob McDonald
- SpaceX rocket explosion debris likely found by UK Coast Guard

SpaceX has another launch under its belt, but not another rocket landing.

The leftover first-stage booster hit the floating platform hard Friday, said SpaceX chief Elon Musk. The company never expected to nail this landing, he said, because of the faster speed of the booster that was required to deliver the satellite to an extra-high orbit.

- **SpaceX pushes satellite launch, rocket landing to Friday**

SpaceX scored a rocket landing on the ground at Cape Canaveral in December, but has yet to nail a trickier barge landing at sea.

The good news, though, is that the unmanned Falcon 9 rocket successfully hoisted the broadcasting satellite for Luxembourg-based company SES.

It was the fifth launch attempt over the past 1½ weeks; Sunday's try ended with an engine shutdown a split second before liftoff. Friday's sunset launch provided a stunning treat along the coast.

SPACEX LAUNCHES SATELLITE, BUT FAILS TO LAND ROCKET ON BARGE

AP

Space-X's Falcon 9 rocket with the Jason-3 satellite aboard, stands ready for flight at Vandenberg Air Force Base, Calif. on Saturday, Jan. 16, 2016. (Matt Hartman)

Share G+1 Tweet

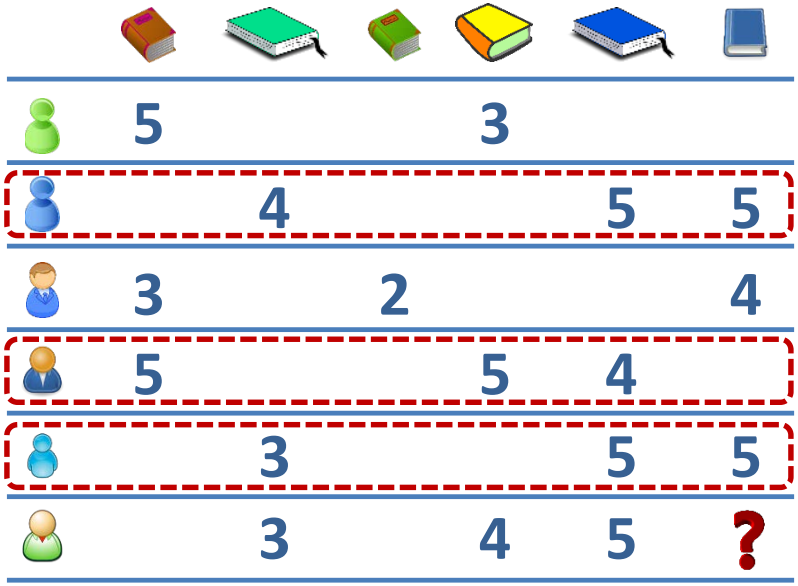
AP

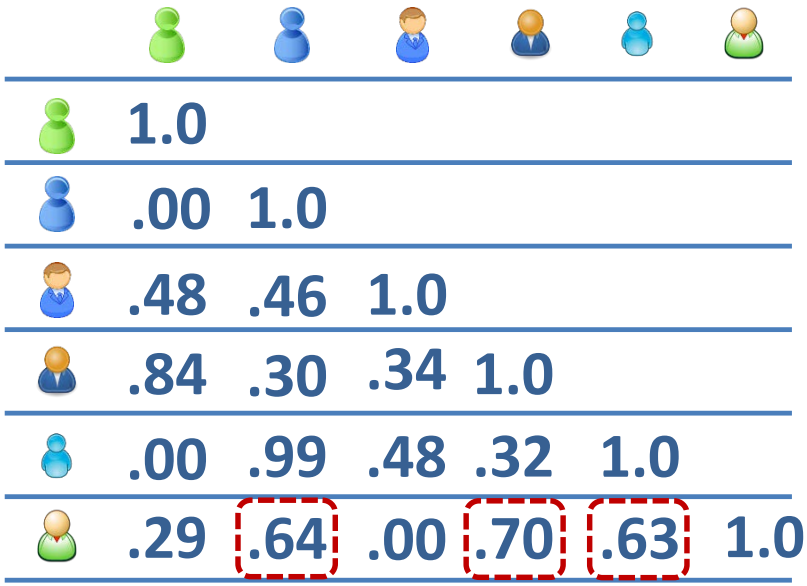
Saturday, March 05, 2016 02:24PM

CAPE CANAVERAL, Fla. -- SpaceX has another launch under its belt, but not another rocket landing.

The leftover first-stage booster hit the floating platform hard Friday, said SpaceX chief Elon Musk. The company never expected to nail this landing, he said, because of the faster speed of the booster that was required to deliver the satellite to an extra-high orbit.

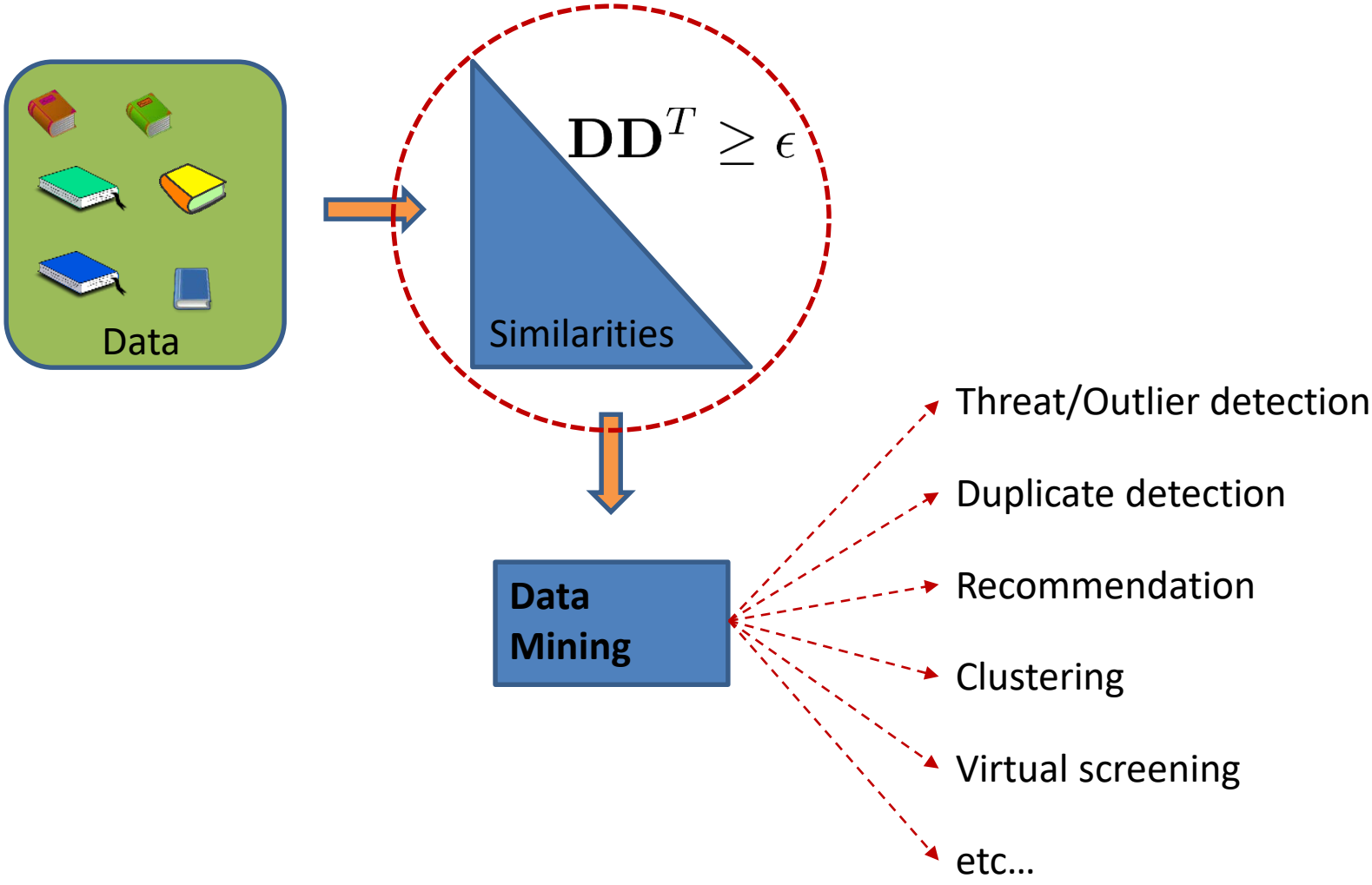
Collaborative Filtering





★★★★★ Loved it
★★★★☆ Liked it
★★★☆☆ It was ok
★★☆☆☆ Disliked it
★☆☆☆☆ Hated it

? = 5


Data
$\mathbf{DD}^T \geq \epsilon$
Similarities
Data Mining
Threat/Outlier detection
Duplicate detection
Recommendation
Clustering
Virtual screening
etc…

Outline

- Nearest neighbor (NN) search
  - IdxJoin – a straightforward solution
  - TAPNN – Tanimoto All-Pairs similarity search
  - L2AP – Cosine All-Pairs similarity search *(brief)*
  - pL2AP – Parallel All-Pairs similarity search
  - Distributed similarity graph construction
- Ongoing and future work

The problem

- For each object d_i from a set D,

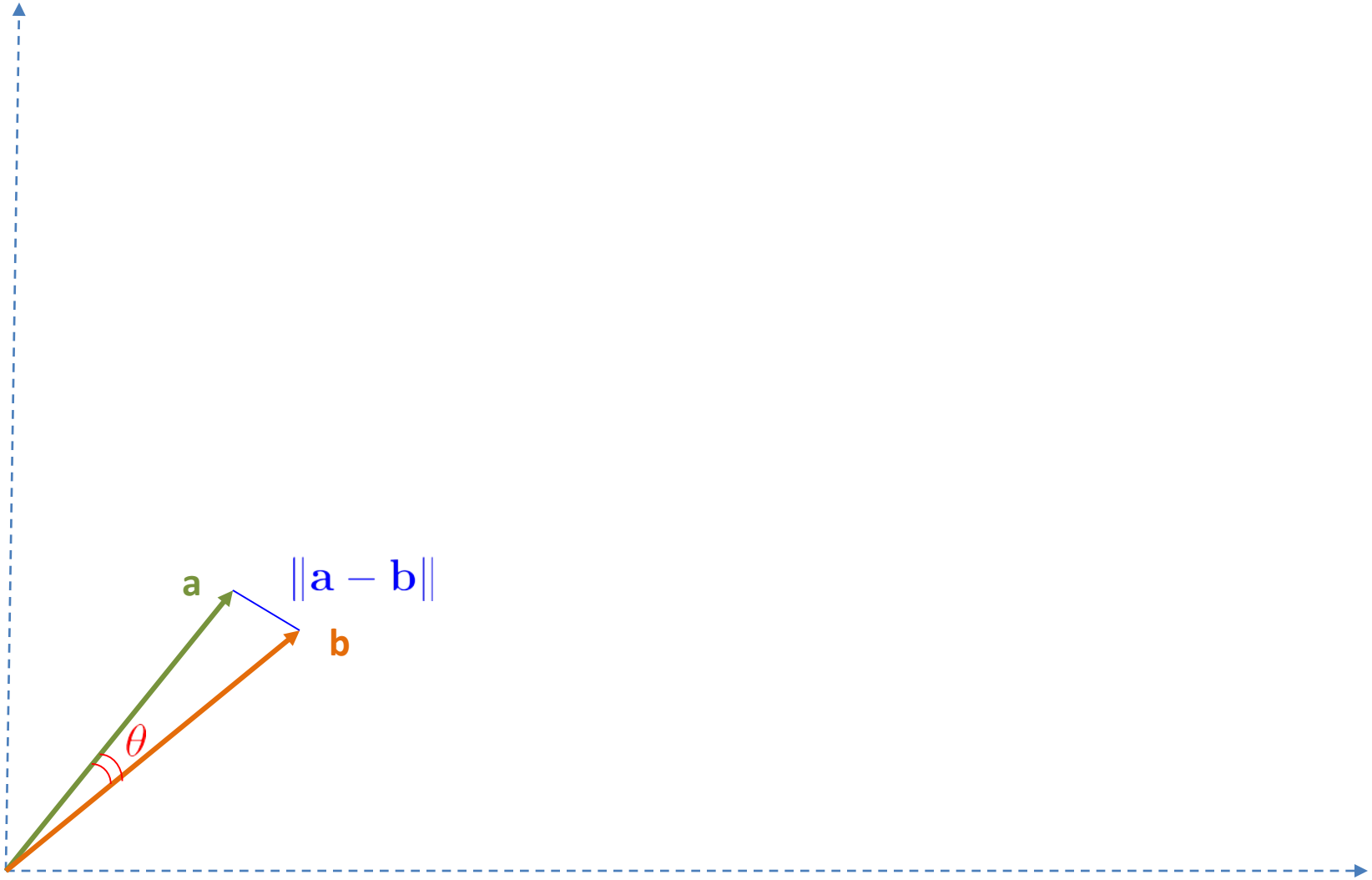
  find all neighbors d_j with $\mathrm{T}(d_i, d_j) \geq \epsilon$.

$$\mathrm{T}(d_i, d_j) = \frac{\langle \mathbf{d}_i, \mathbf{d}_j \rangle}{||\mathbf{d}_i||_2^2 + ||\mathbf{d}_j||_2^2 - \langle \mathbf{d}_i, \mathbf{d}_j \rangle}$$

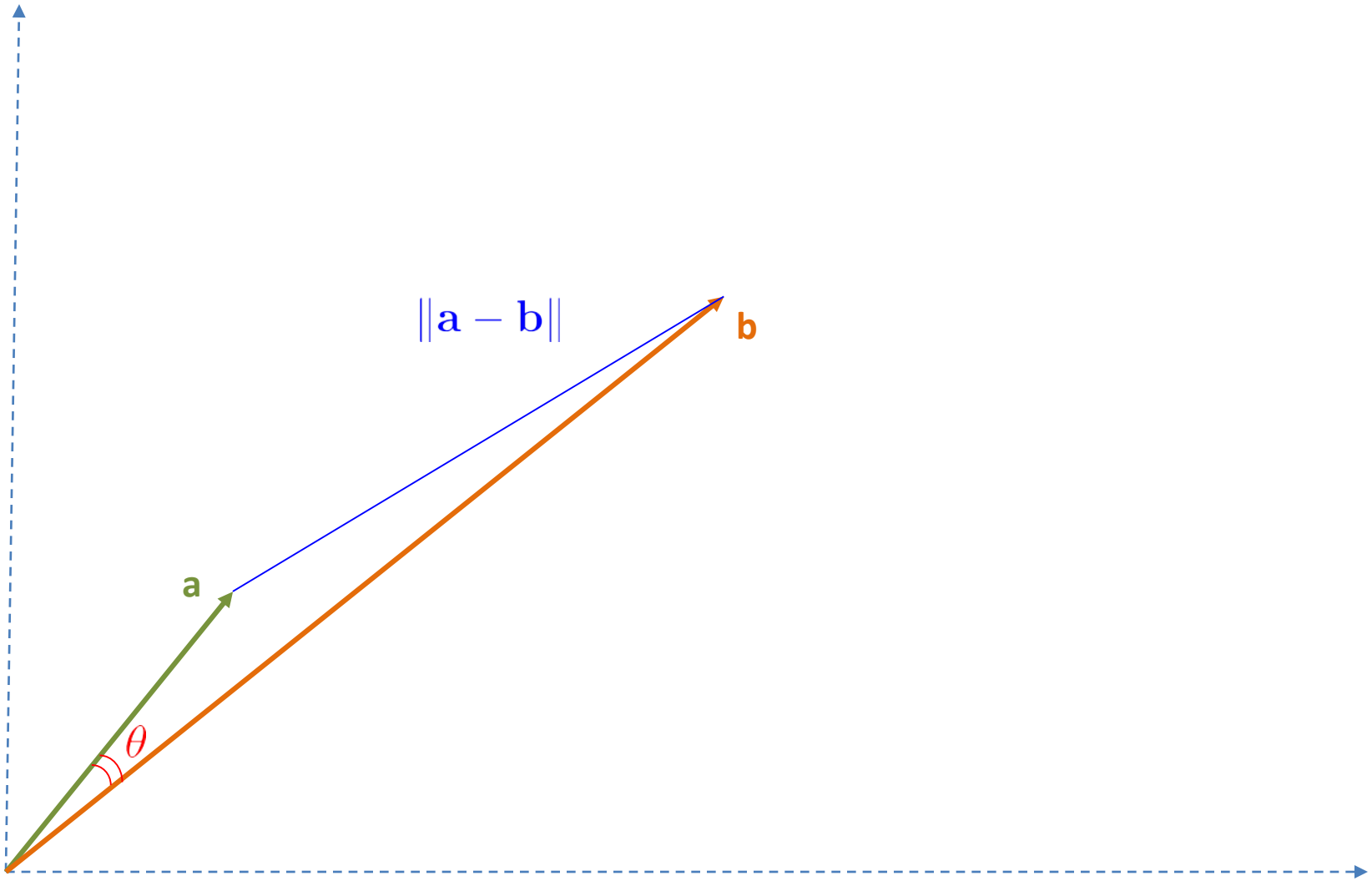
  find all neighbors d_j with $\mathrm{C}(d_i, d_j) \geq \epsilon$.

$$\mathrm{C}(d_i, d_j) = \frac{\langle \mathbf{d}_i, \mathbf{d}_j \rangle}{||\mathbf{d}_i||_2 \times ||\mathbf{d}_j||_2}$$

Why Tanimoto and Cosine?

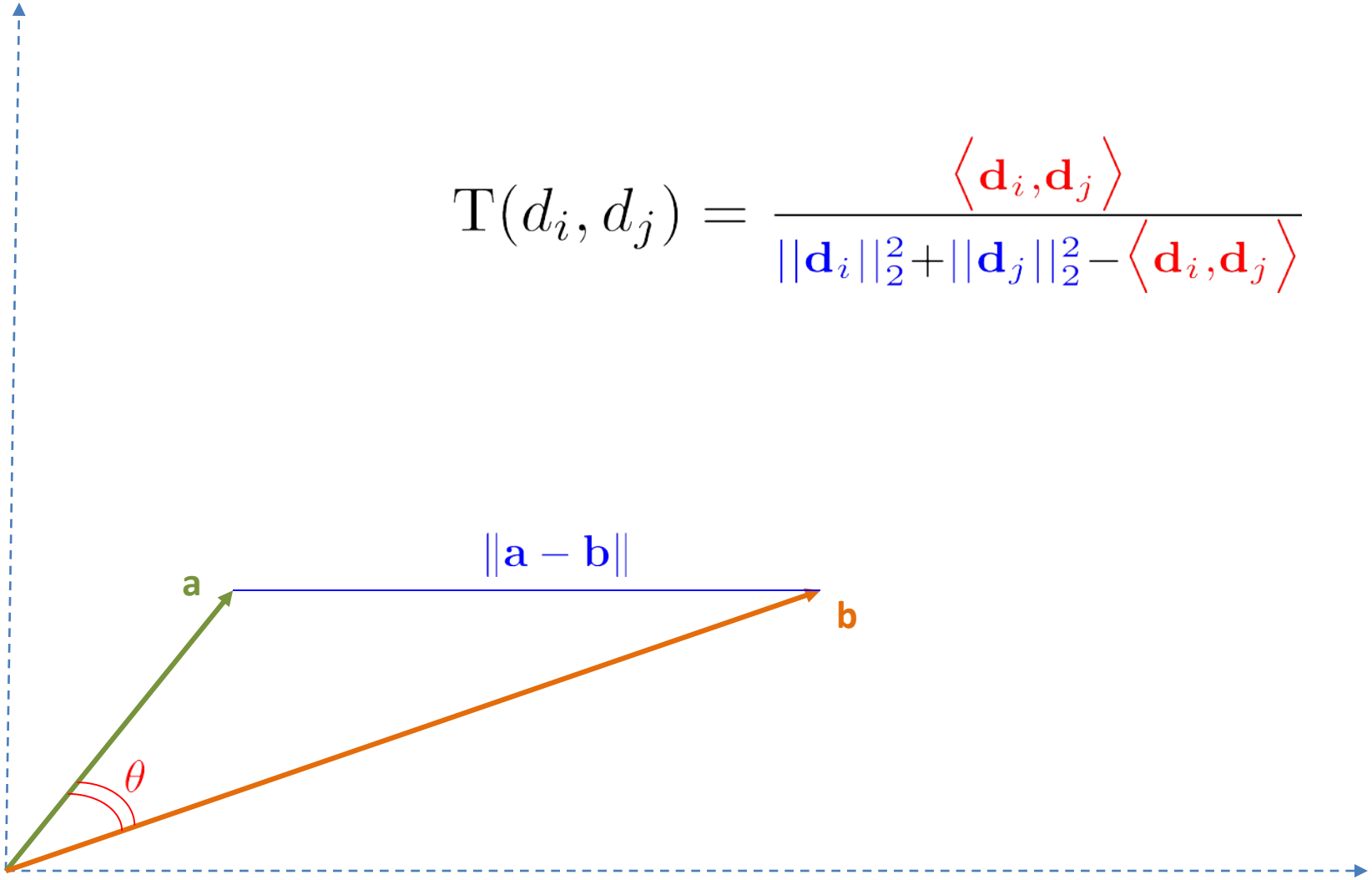


Why Tanimoto and Cosine?



Why Tanimoto and Cosine?

$$\mathrm{T}(d_i, d_j) = \frac{\left\langle \mathbf{d}_i, \mathbf{d}_j \right\rangle}{||\mathbf{d}_i||_2^2 + ||\mathbf{d}_j||_2^2 - \left\langle \mathbf{d}_i, \mathbf{d}_j \right\rangle}$$



Why Tanimoto and Cosine?

- Tanimoto coefficient particularly useful for analyzing sparse asymmetric attribute data

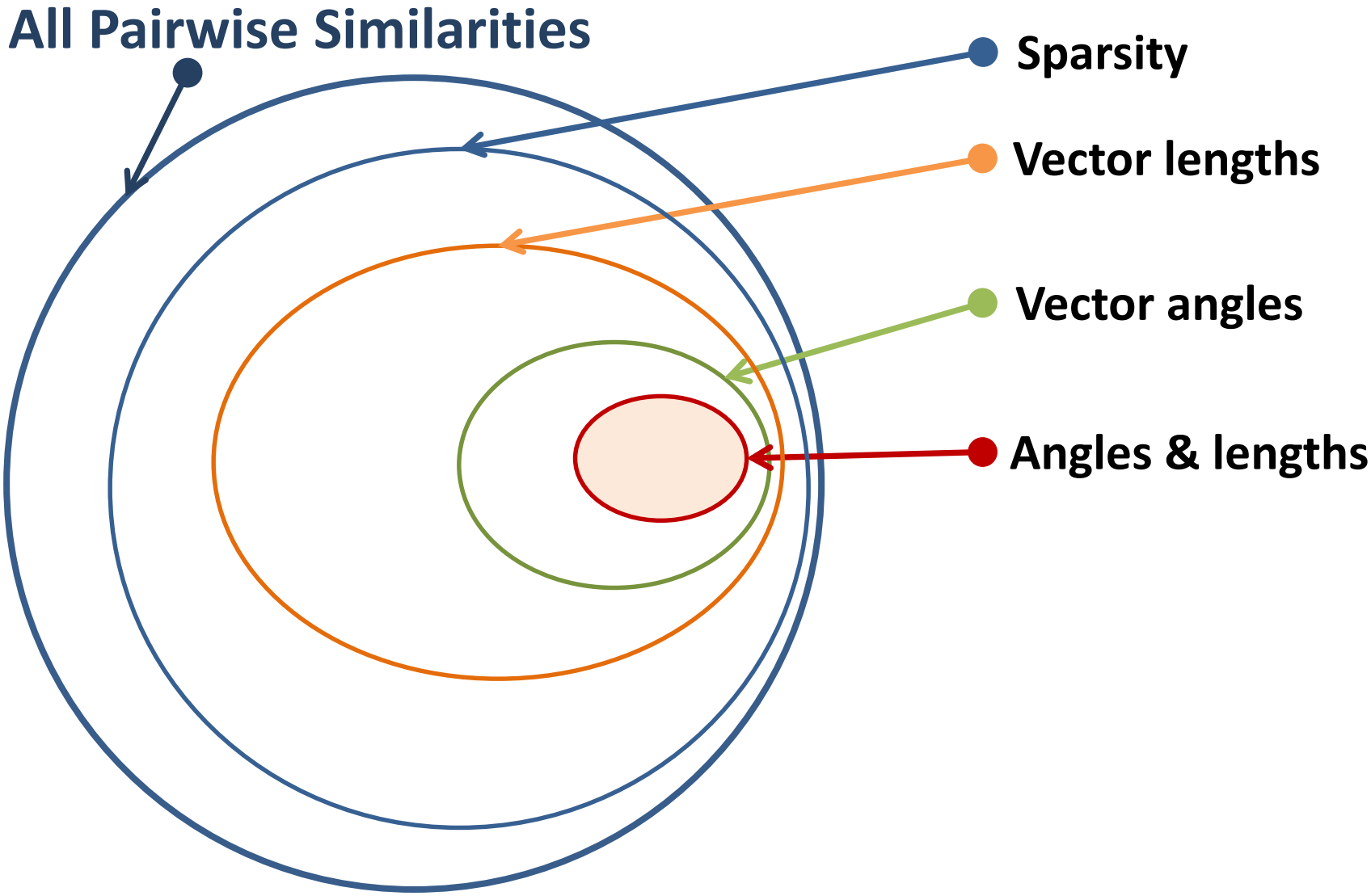
$$H_2O = (2, 1)$$

$$H_4O_2 = (4, 2)$$

$$\text{Cosine}(H_2O, H_4O_2) = 1$$

$$\text{Tanimoto}(H_2O, H_4O_2) = 0.66$$

How: pruning the search space



Outline

- Nearest neighbor (NN) search
  - IdxJoin – a straightforward solution
  - TAPNN – Tanimoto All-Pairs similarity search
  - L2AP – Cosine All-Pairs similarity search *(brief)*
  - pL2AP – Parallel All-Pairs similarity search
  - Distributed similarity graph construction
- Ongoing and future work

IdxJoin: A straight-forward solution

- Method:

  Compute and store vector norms

  Construct an inverted index from the objects

  For each query object:

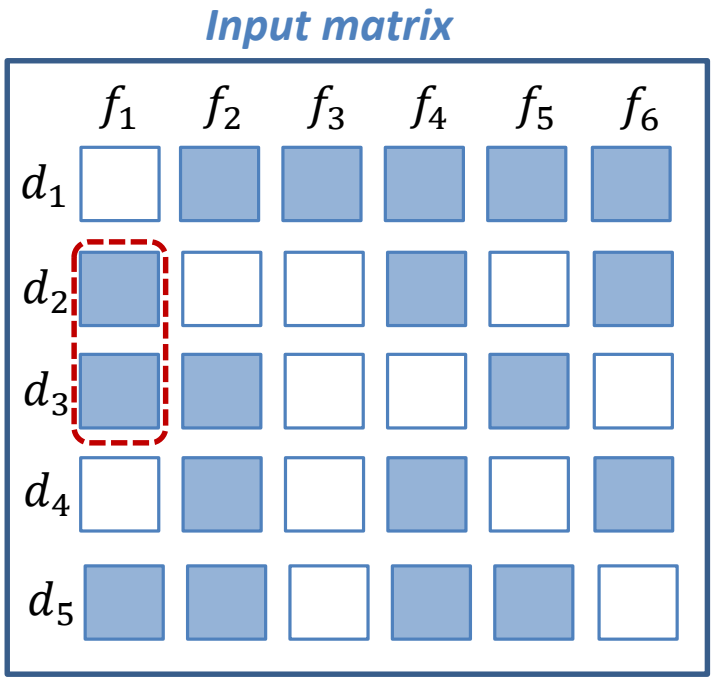
  – Compare only with objects with features in common

  – Select neighbors

- Advantage:
  - Skips some object comparisons and many meaningless multiply-adds

IdxJoin: A straight-forward solution

Main idea: leverage data sparsity



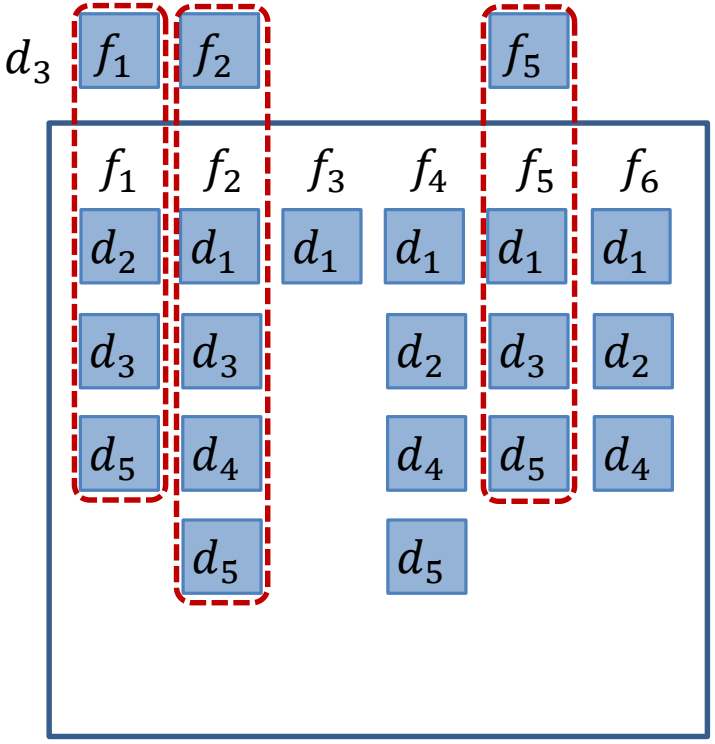
$$\mathrm{T}(d_i, d_j) = \frac{\langle \mathbf{d}_i, \mathbf{d}_j \rangle}{||\mathbf{d}_i||_2^2 + ||\mathbf{d}_j||_2^2 - \langle \mathbf{d}_i, \mathbf{d}_j \rangle}$$

$$\mathrm{C}(d_i, d_j) = \frac{\langle \mathbf{d}_i, \mathbf{d}_j \rangle}{||\mathbf{d}_i||_2 \times ||\mathbf{d}_j||_2}$$

$$\langle \mathbf{d}_2, \mathbf{d}_3 \rangle = d_{2,1} \times d_{3,1} + d_{2,2} \times d_{3,2} + d_{2,3} \times d_{3,3} + d_{2,4} \times d_{3,4} + d_{2,5} \times d_{3,5} + d_{2,6} \times d_{3,6}$$

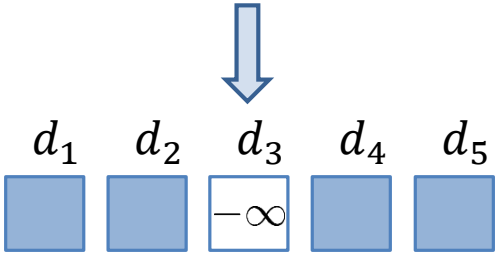
IdxJoin: Accumulation

- Inverted index: set of lists, one for each feature, containing documents and their associated values



Inverted Index

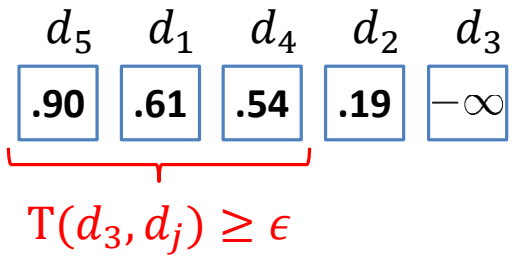
$$A[d_2] \mathrel{+}= d_{3,1} \times d_{2,1}$$
$$A[d_5] \mathrel{+}= d_{3,1} \times d_{5,1}$$
$$A[d_1] \mathrel{+}= d_{3,2} \times d_{1,2}$$
$$A[d_4] \mathrel{+}= d_{3,2} \times d_{4,2}$$
$$A[d_5] \mathrel{+}= d_{3,2} \times d_{5,2}$$
$$[\ldots]$$



Accumulator

IdxJoin: Neighbor selection

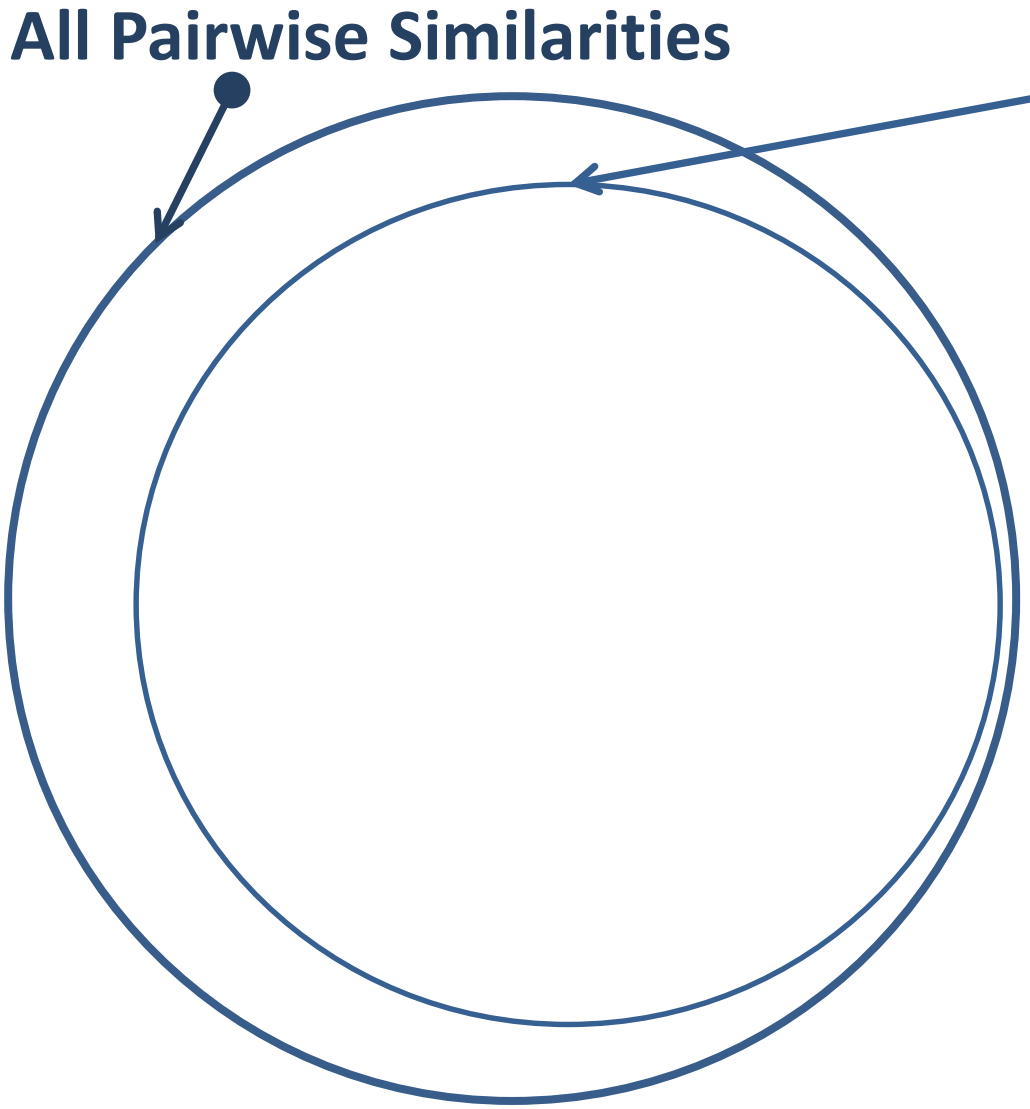
- Compute and store vector norms
- Compute dot-products with candidates in index
- Apply $\mathrm{T}(d_q, d_c)$ or $\mathrm{C}(d_q, d_c)$ formula
- Sort the resulting similarities
- Restrict neighbors:
  - Keep those candidates with $\mathrm{sim}(d_q, d_c) \geq \epsilon$.



Outline

- Nearest neighbor (NN) search
  - IdxJoin – a straightforward solution
  - TAPNN – Tanimoto All-Pairs similarity search
  - L2AP – Cosine All-Pairs similarity search *(brief)*
  - pL2AP – Parallel All-Pairs similarity search
  - Distributed similarity graph construction
- Ongoing and future work

How: pruning the search space



Can we do better?

Leverage

vector lengths

Length based pruning

- d_q and d_c cannot be neighbors unless

$$\|\mathbf{d}_c\| \in [(1/\alpha)\|\mathbf{d}_q\|,\ \alpha\|\mathbf{d}_q\|]\,,$$

$$\alpha = \frac{1}{2}\left(\left(1+\frac{1}{\epsilon}\right) + \sqrt{\left(1+\frac{1}{\epsilon}\right)^2 - 4}\right)$$

  - α bound due to Marzena Kryszkiewicz, IIDS 2013

- If we process objects in non-decreasing length order, we only need to check

$$\|\mathbf{d}_q\| \geq \frac{1}{\alpha}\|\mathbf{d}_c\|$$

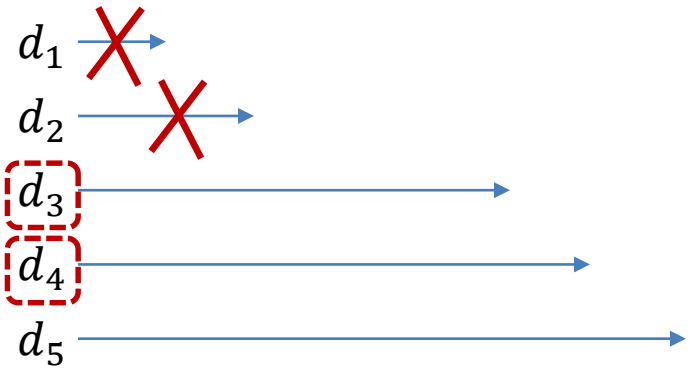
Length based pruning

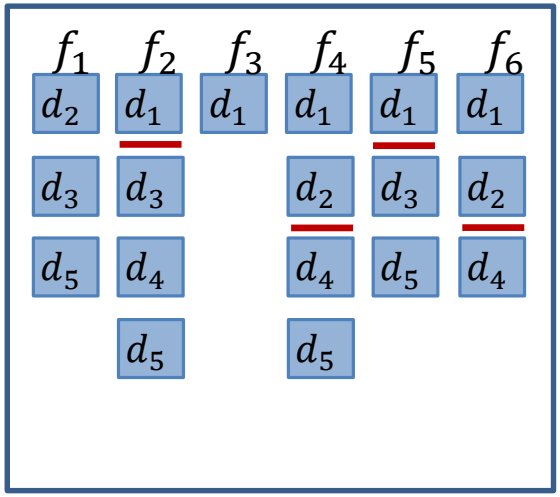
- d_q and d_c cannot be neighbors unless

$$\|\mathbf{d}_c\| \in [(1/\alpha)\|\mathbf{d}_q\|,\ \alpha\|\mathbf{d}_q\|]\,,$$

$$\alpha = \frac{1}{2}\left(\left(1+\frac{1}{\epsilon}\right) + \sqrt{\left(1+\frac{1}{\epsilon}\right)^2 - 4}\right)$$

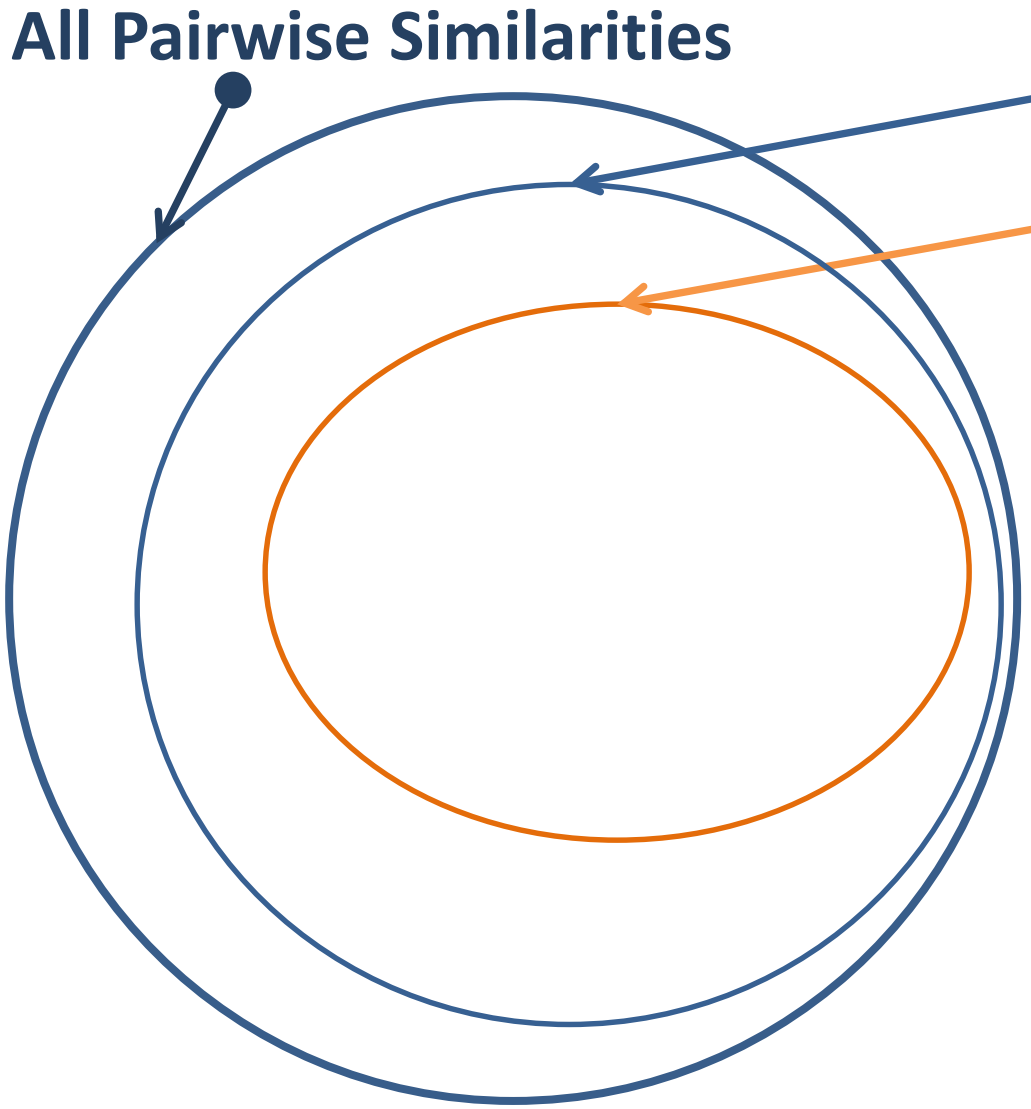
  - α bound due to Marzena Kryszkiewicz, IIDS 2013
- Relabel objects in non-decreasing length order





Inverted Index

How: pruning the search space



Can we do better?

Leverage

vector angles

Subset of cosine neighborhood

- The following inequalities hold for our domain:

$$\mathrm{T}(d_i, d_j) \leq \mathrm{C}(d_i, d_j)$$
$$\mathrm{T}(d_i, d_j) \geq \epsilon \Rightarrow \mathrm{C}(d_i, d_j) \geq \epsilon$$
$$\mathrm{C}(d_i, d_j) < \epsilon \Rightarrow \mathrm{T}(d_i, d_j) < \epsilon$$

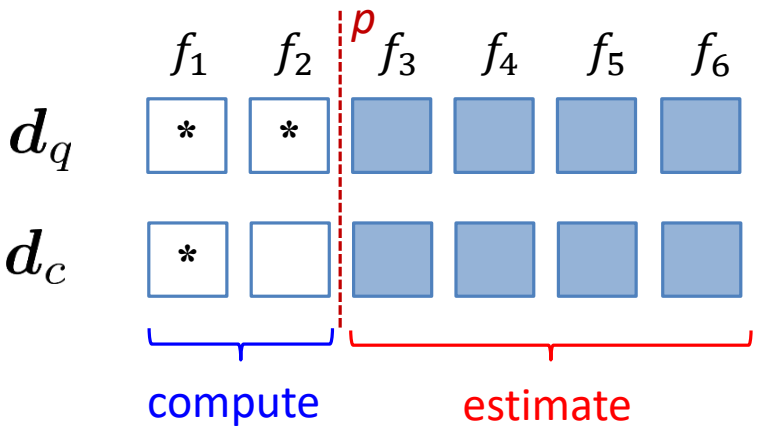
- Potential solution
  - Store vector norms and normalize vectors
  - Find cosine neighbors
  - Transform $\mathrm{C}(d_i, d_j)$ to $\mathrm{T}(d_i, d_j)$
  - Remove non-Tanimoto neighbors
- Tighter bound due to Lee et al., DEXA 2010

$$\mathrm{T}(d_i, d_j) \geq \epsilon \Rightarrow \mathrm{C}(d_i, d_j) \geq \frac{2\epsilon}{1+\epsilon} = t$$

L2AP: Fast cosine similarity search

Main idea: leverage similarity estimates

$$\langle \mathbf{d}_q, \mathbf{d}_c \rangle = \langle \mathbf{d}_q^{\leq p}, \mathbf{d}_c^{\leq p} \rangle + \langle \mathbf{d}_q^{>p}, \mathbf{d}_c^{>p} \rangle$$

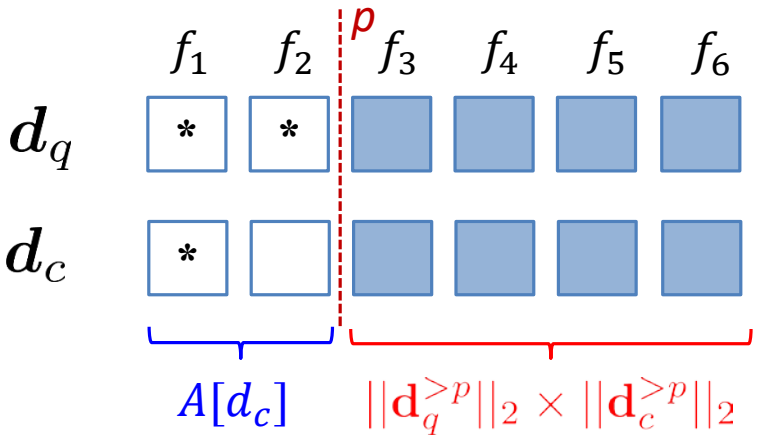


Angle Filtering

Filter/prune object pairs not in final graph based on *similarity estimates*

$$\left\langle \mathbf{d}_q^{>p}, \mathbf{d}_c^{>p} \right\rangle \leq ||\mathbf{d}_q^{>p}||_2 \times ||\mathbf{d}_c^{>p}||_2$$

(Cauchy-Schwarz inequality)

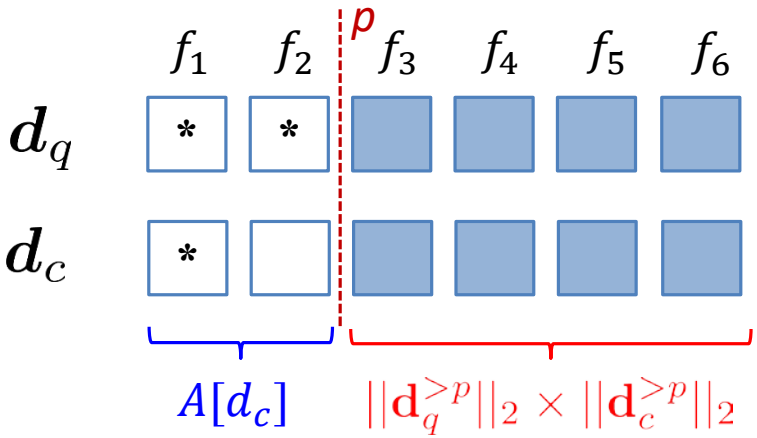


Filter $\mathrm{T}(d_q, d_c)$ if $A[d_c] + ||\mathbf{d}_q^{>p}||_2 \times ||\mathbf{d}_c^{>p}||_2 < t$

Angle Filtering

Filter/prune object pairs not in final graph based on *similarity estimates*

$$\left\langle \mathbf{d}_q^{>p}, \mathbf{d}_c^{>p} \right\rangle \leq \min(||\mathbf{d}_q^{>p}||_0, ||\mathbf{d}_c^{>p}||_0) \times ||\mathbf{d}_q^{>p}||_\infty \times ||\mathbf{d}_c^{>p}||_\infty$$



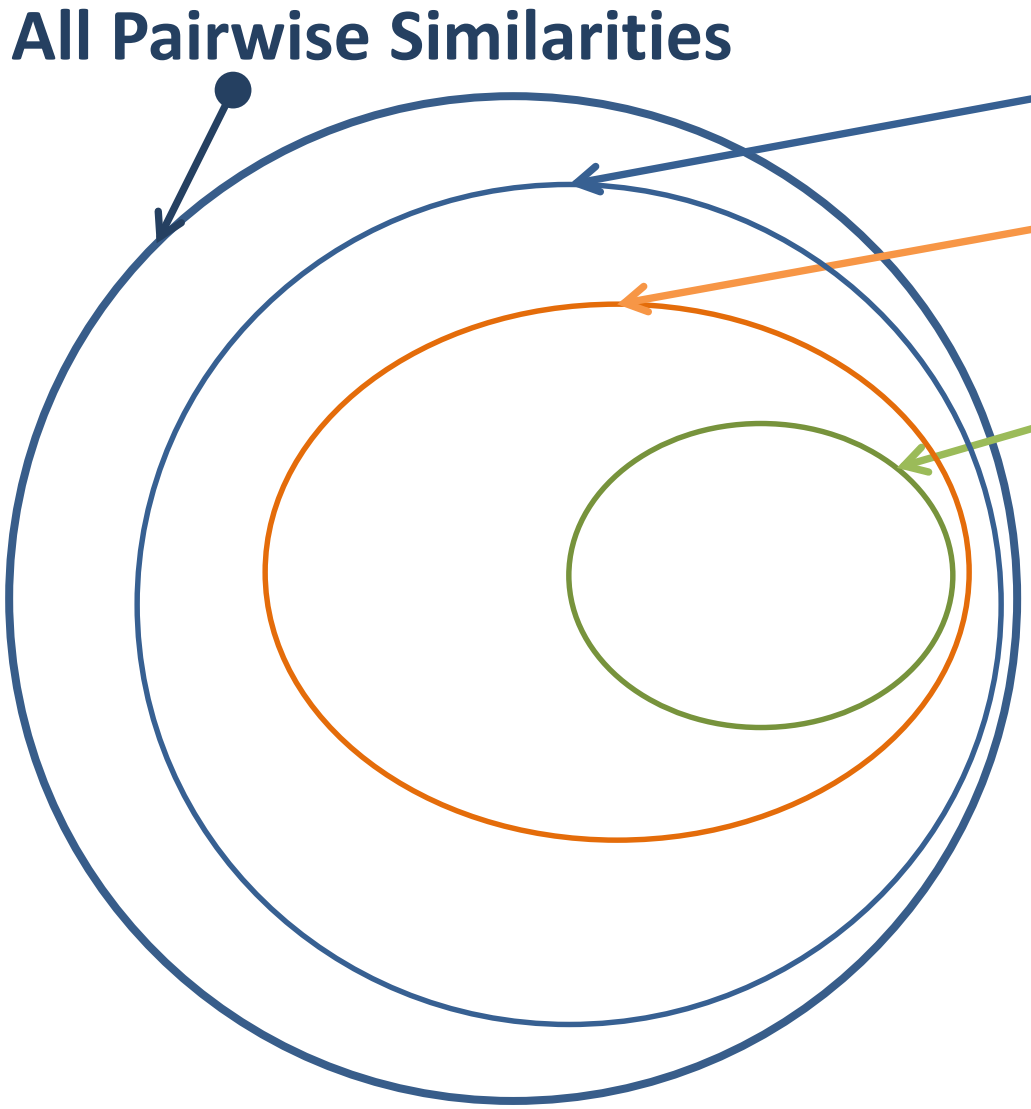
Filter $\mathrm{T}(d_q, d_c)$ if

$$A[d_c] + \min(||\mathbf{d}_q^{>p}||_0, ||\mathbf{d}_c^{>p}||_0) \times ||\mathbf{d}_q^{>p}||_\infty \times ||\mathbf{d}_c^{>p}||_\infty > t$$

TAPNN: Filtering

- L2AP filtering contingent on processing order
- Developed order-agnostic filtering bounds
- Integrated vector length-based index skip-pointers

How: pruning the search space



Can we do better?

Leverage vector angles & their lengths

Angle + length

- d_q and d_c cannot be neighbors unless

$$\|\mathbf{d}_c\| \in [(1/\alpha)\|\mathbf{d}_q\|,\ \alpha\|\mathbf{d}_q\|]\,,$$

$$\alpha = \frac{1}{2}\left(\left(1+\frac{1}{\epsilon}\right) + \sqrt{\left(1+\frac{1}{\epsilon}\right)^2 - 4}\right)$$

- α is an upper limit of

$$\beta = \frac{s}{t} + \sqrt{\left(\frac{s}{t}\right)^2 - 1}$$

where s is any cosine similarity upper bound such as the ones we compute during filtering

Tanimoto All-Pairs Nearest Neighbors

- TAPNN:

  Compute and store vector norms

  Normalize vectors

  Construct a partial inverted index from the objects

  For each query object:

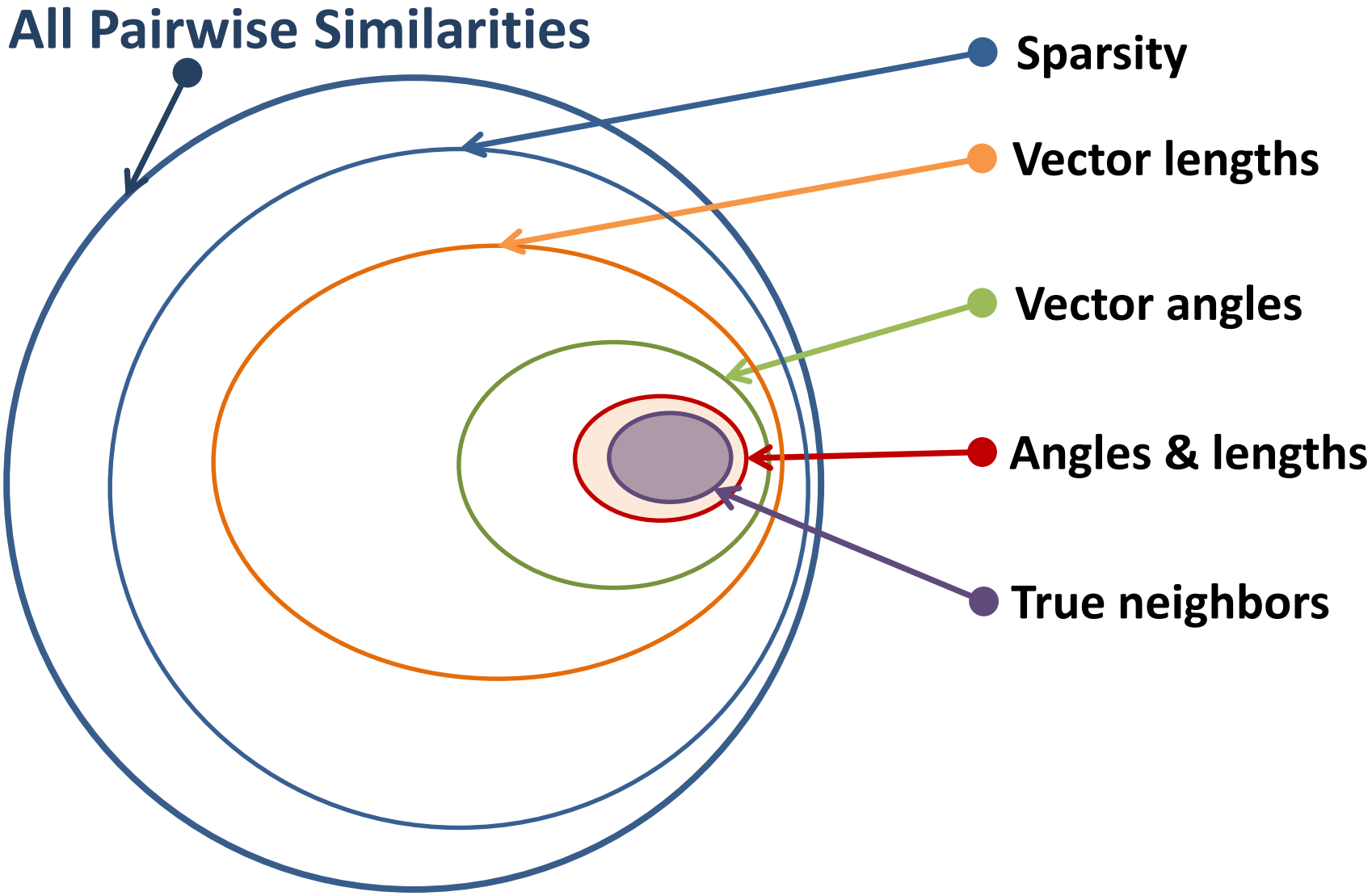
  – Skip short candidates

  – Use cosine similarity upper bounds to filter cosine non-neighbors

  – Use cosine similarity upper bounds to filter Tanimoto non-neighbors

  – Select neighbors

How: pruning the search space

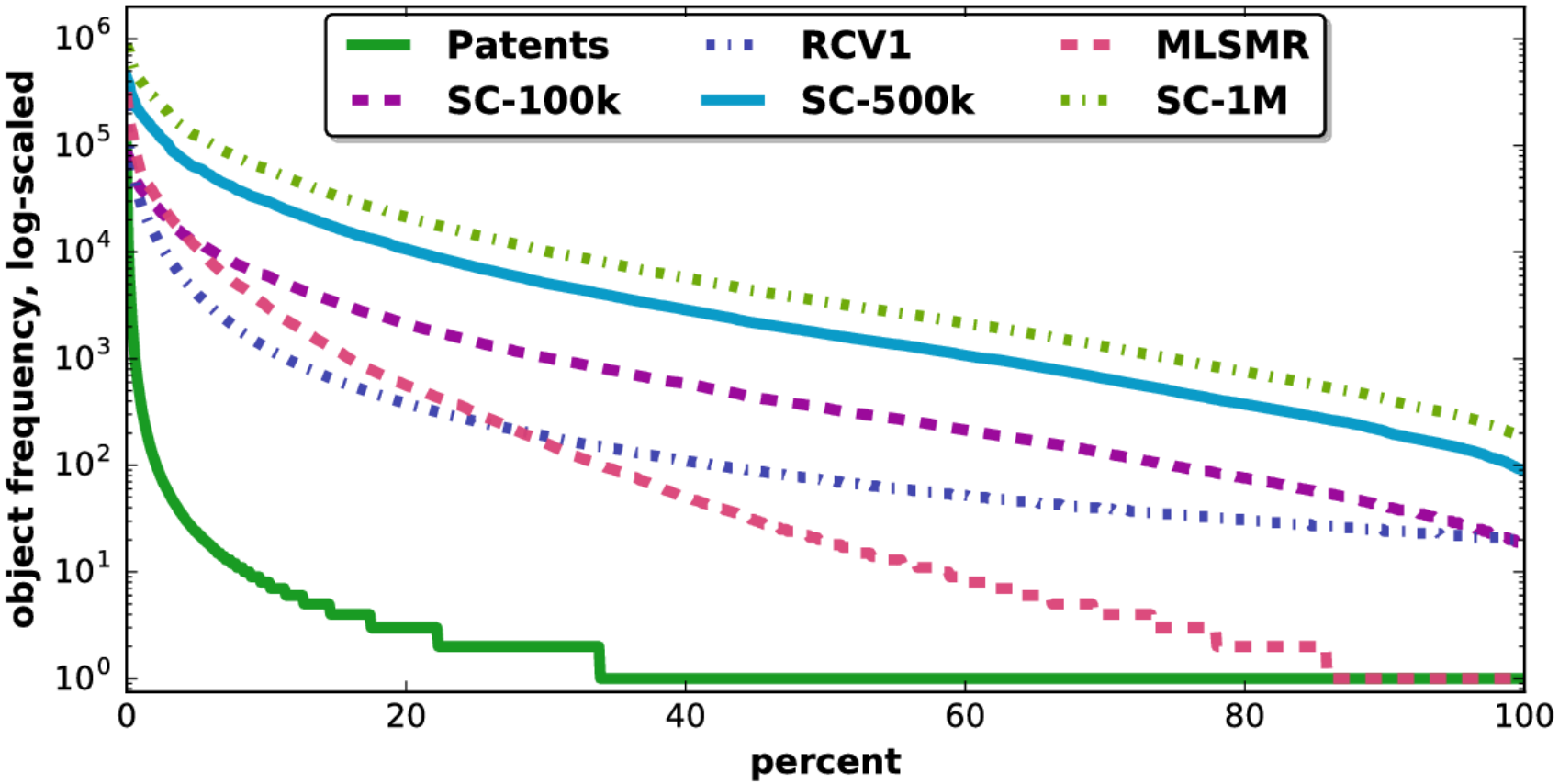


Experimental evaluation: datasets

- Patents: text of random USPTO patents
- RCV1: text of newswire stories
- MLSMR: structures of PubChem compounds
- SC: compounds form the SureChEMBL database

dataset	# objects	# features	# non-zeros
Patents	100,000	759,044	46.3M
RCV1	804,414	45,669	61.5M
MLSMR	325,164	20,021	56.1M
SC	11,519,370	7,415	1,784.5M
SC-5M	5,000,000	7,415	699.9M
SC-1M	1,000,000	6,752	154.9M
SC-500k	500,000	6,717	77.5M
SC-100k	100,000	6,623	15.5M

Experimental evaluation: datasets

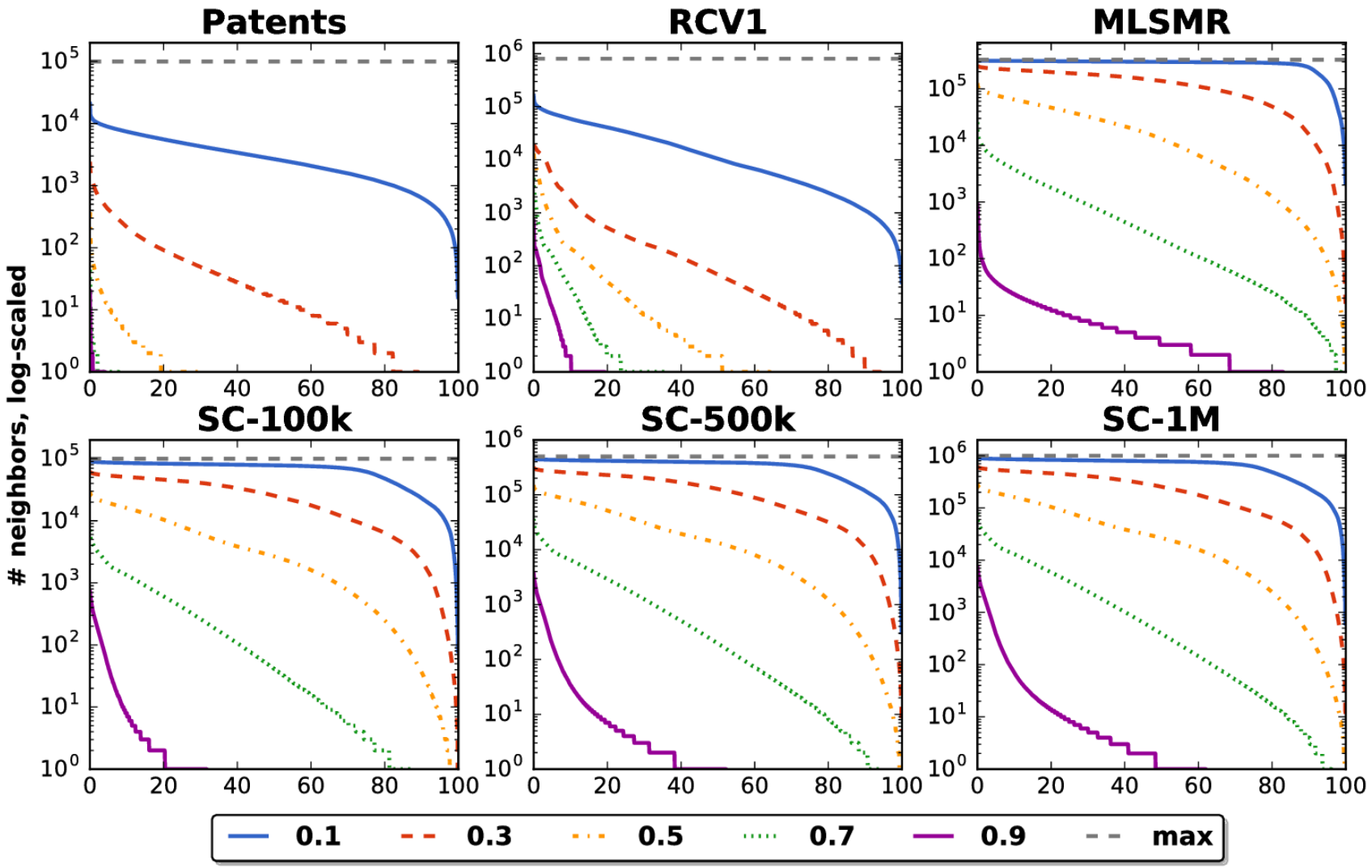


Experimental evaluation: methods

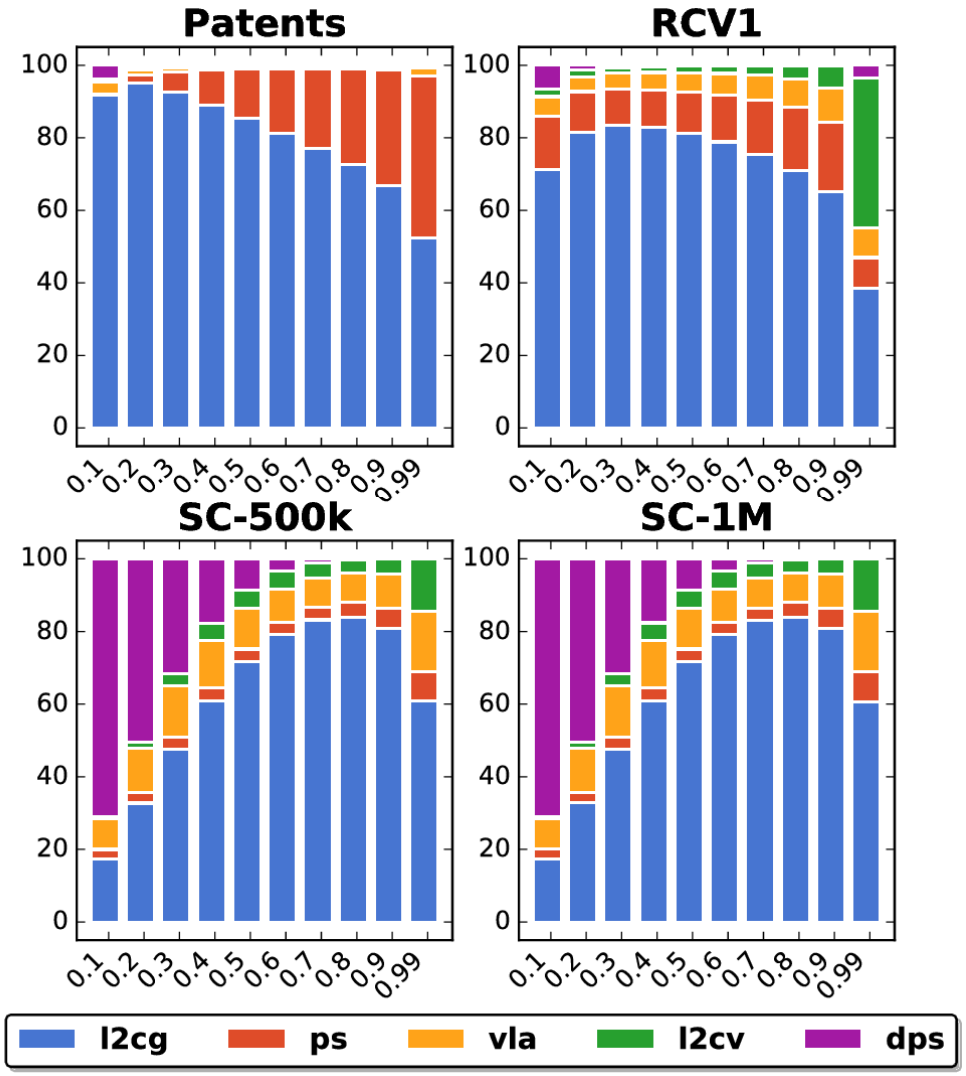
- IdxJoin
- L2AP
- MMJoin
  - Method by Lee et al.
  - Angle based filtering with tighter t bound
- MK-Join
  - Algorithm designed using theoretic bounds by Marzena Kryszkiewicz
  - Method uses same fast accumulator as TAPNN

http://davidanastasiu.net/software/tapnn/

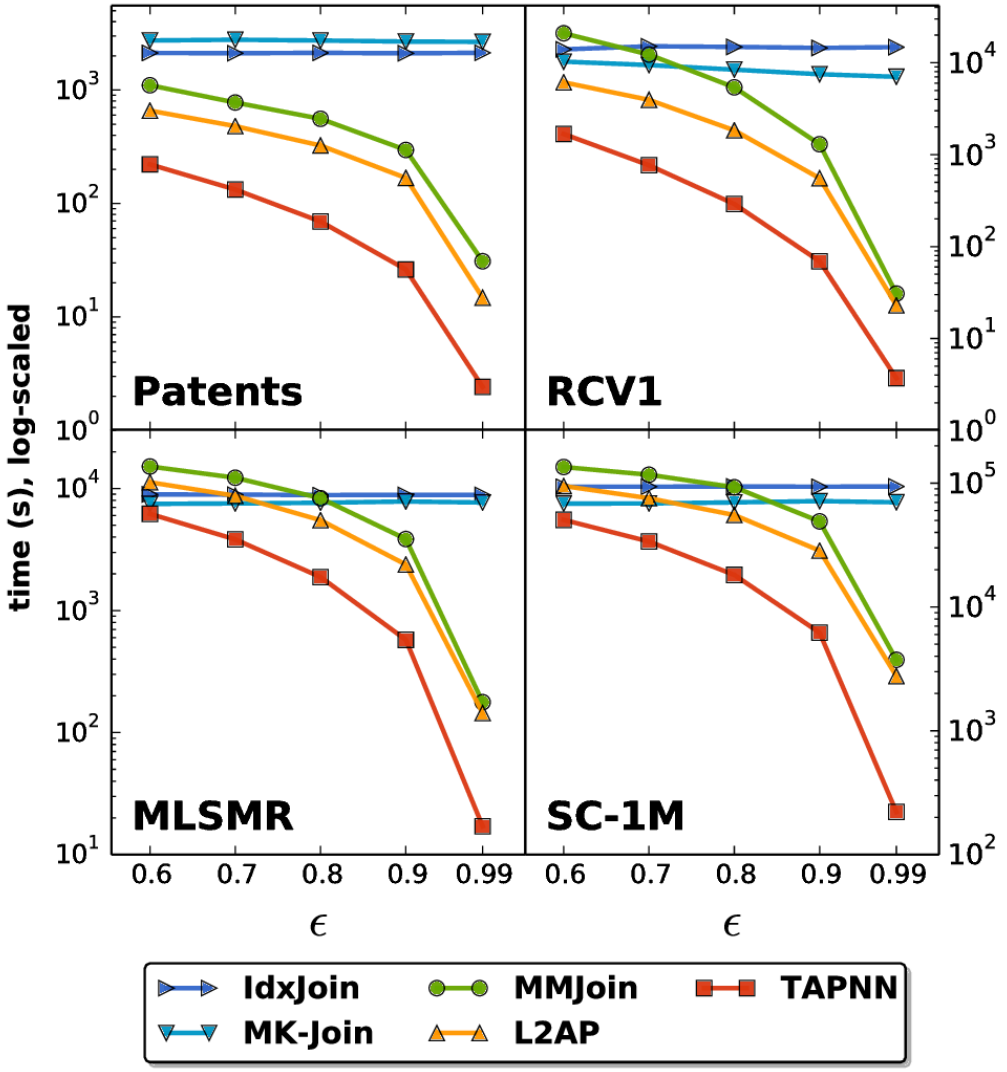
TAPNN results: neighborhood size



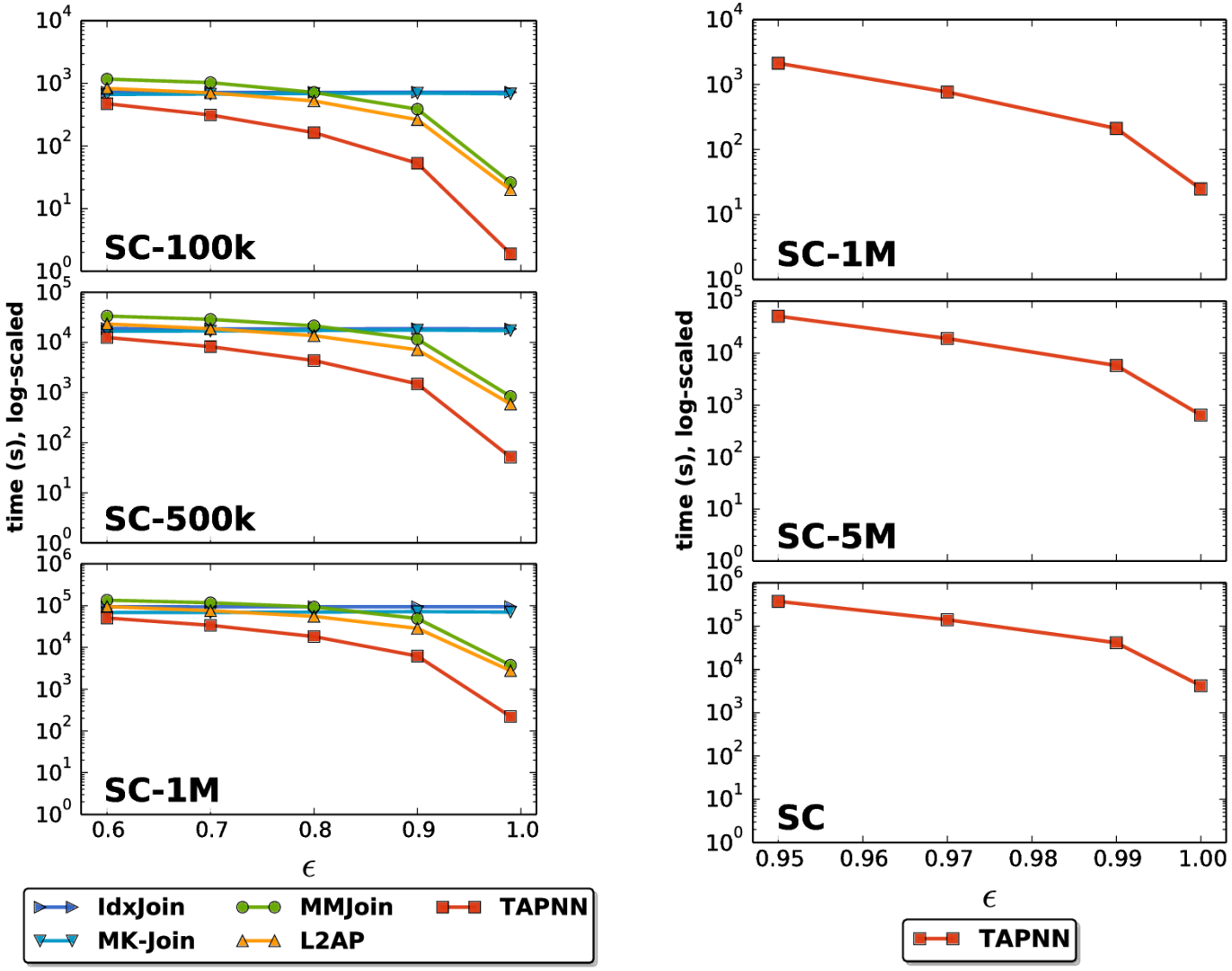
TAPNN results: pruning effectiveness



TAPNN results: efficiency comparison



TAPNN results: scaling



Execution time scaling given increasing problem size.

Outline

- Nearest neighbor (NN) search
  - IdxJoin – a straightforward solution
  - TAPNN – Tanimoto All-Pairs similarity search
  - L2AP – Cosine All-Pairs similarity search
  - pL2AP – Parallel All-Pairs similarity search
  - Distributed similarity graph construction
- Ongoing and future work

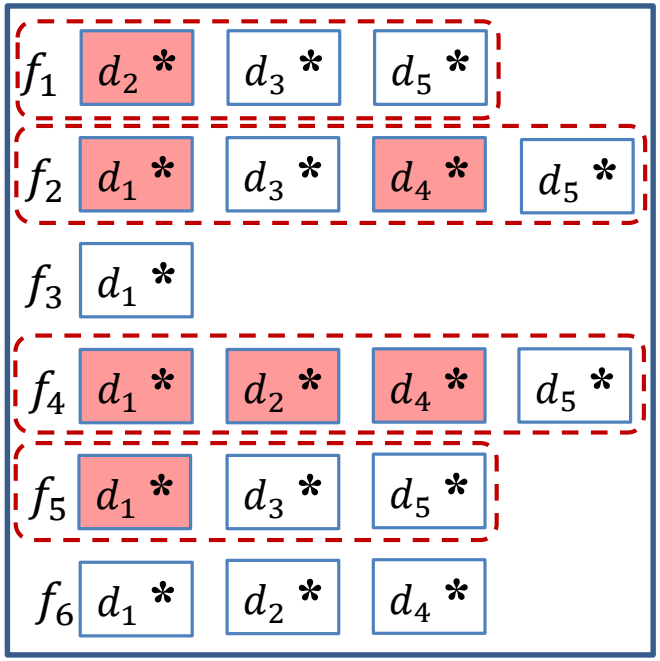
L2AP: Filtering

Could have many useless memory operations

If d_1, d_2, d_4 were pruned

d_1	d_2	d_3	d_4	d_5
-1	-1	.25	-1	.54

Accumulator

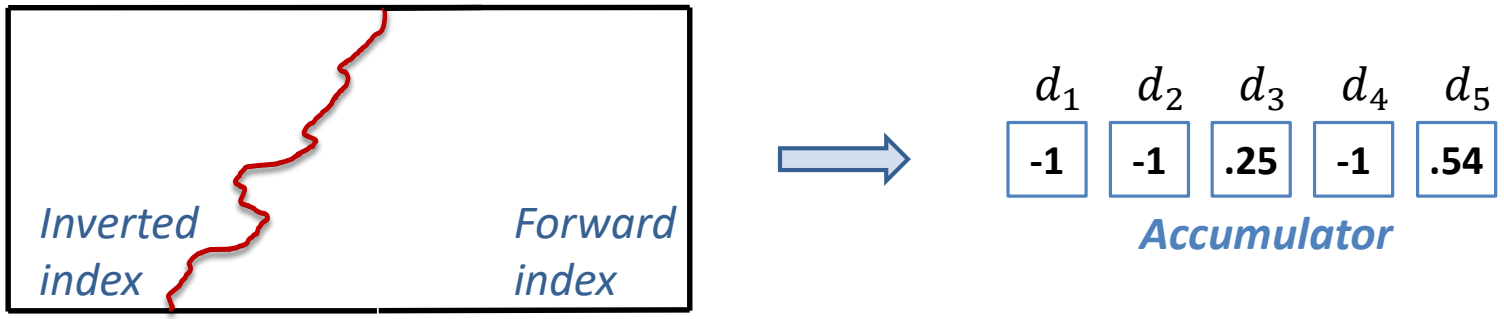


Inverted index

L2AP: algorithm

L2AP follows a two-step process:

1. Accumulate similarity using partial inverted index



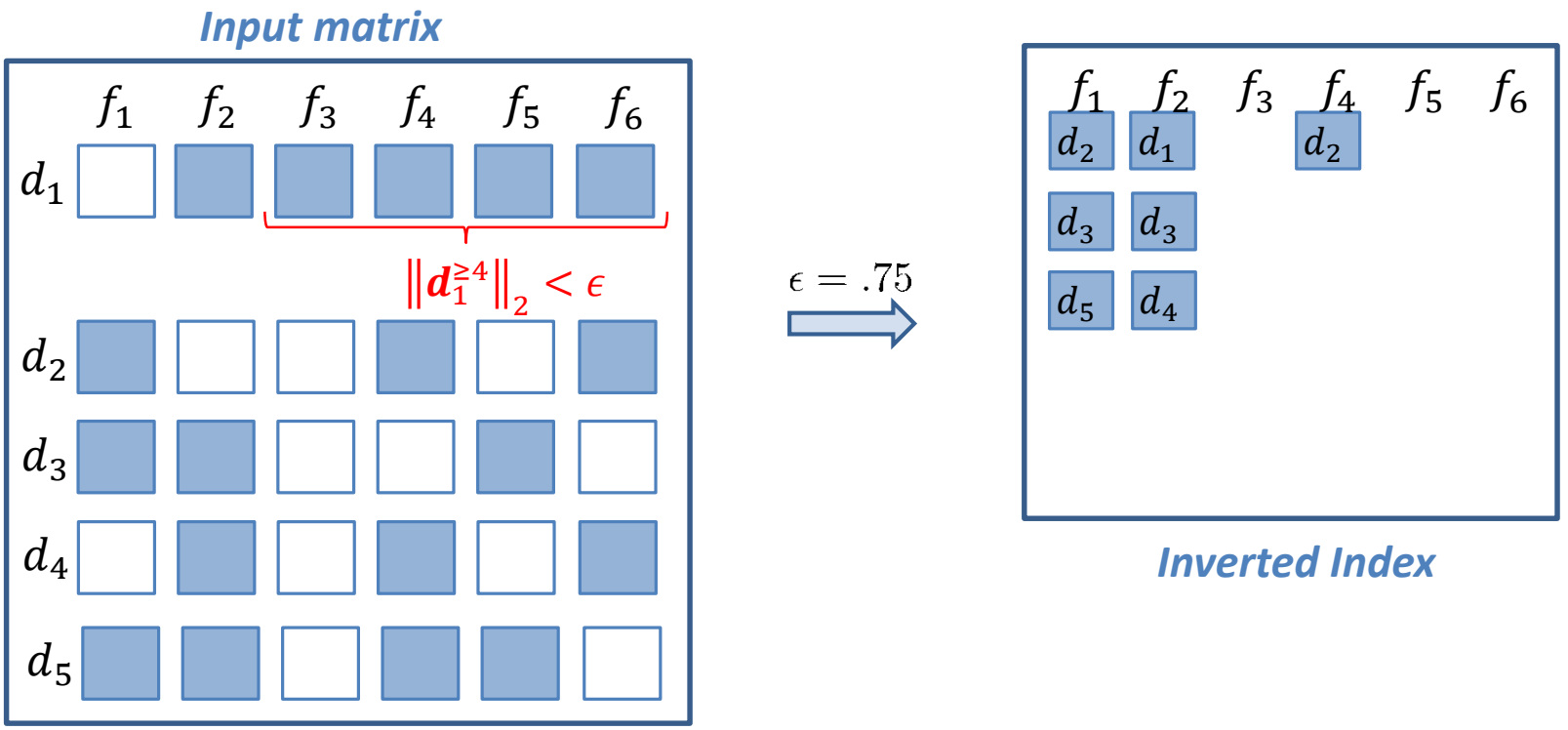
2. For each un-pruned object, finish similarity computation using forward index
   - Only need to compute $\text{sim}(d_q, d_3^>)$ & $\text{sim}(d_q, d_5^>)$
   - Can do further filtering

L2AP: Indexing

Solution:

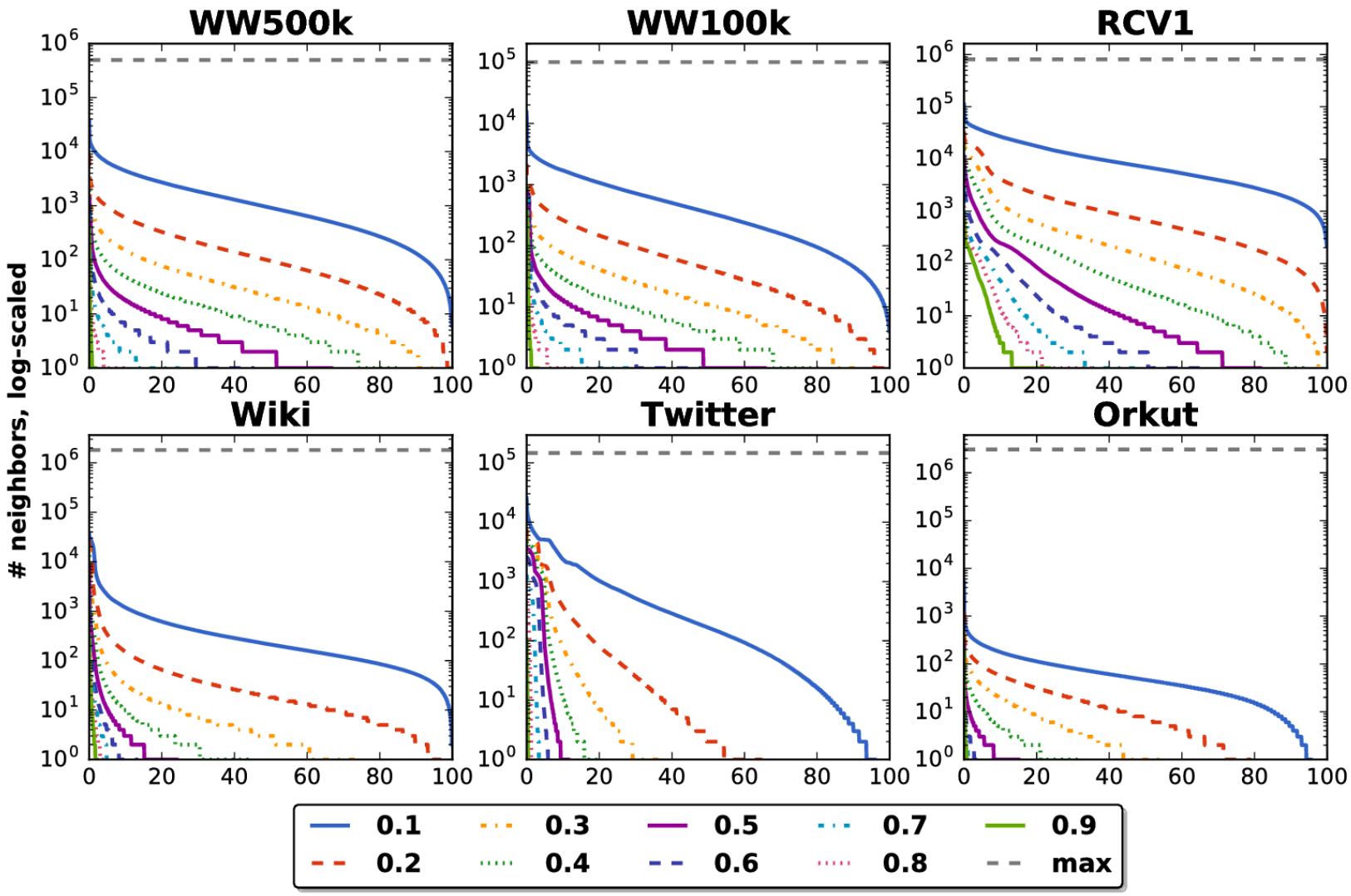
Index enough non-zeros to guarantee correct result.

Same idea used to stop accepting new candidates during CG.



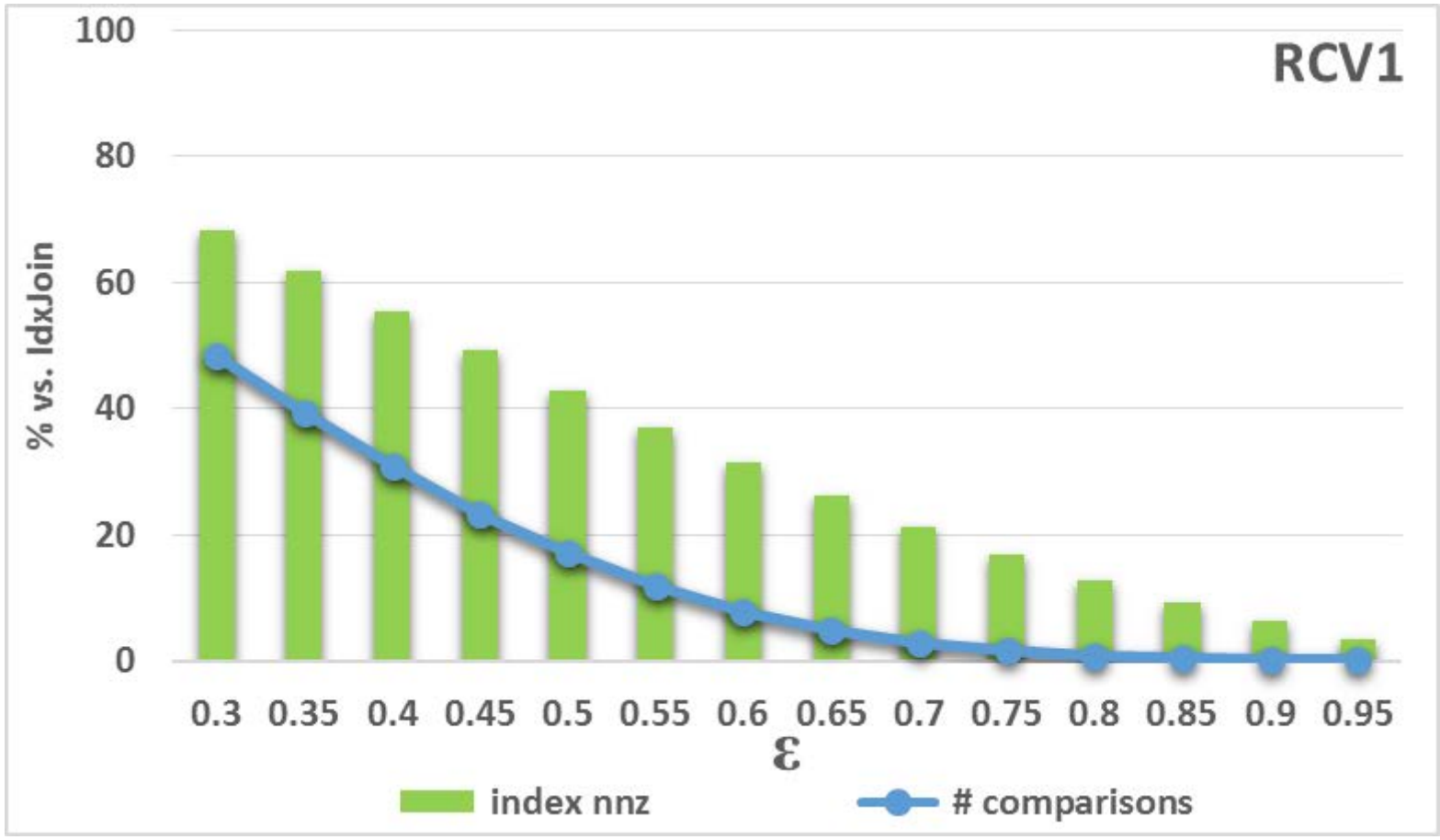
L2AP: Datasets

Dataset	n	m	nnz	mrl	mcl
RCV1	804,414	43,001	61e6	76	1417
WW500	494,244	343,622	197e6	399	574
WW100	100,528	339,944	79e6	787	233
Twitter	146,170	143,469	200e6	1370	1395
Wiki	1,815,914	1,648,879	44e6	24	27
Orkut	3,072,626	3,072,441	223e6	73	73


WW500k
WW100k
RCV1
Wiki
Twitter
Orkut
neighbors, log-scaled
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
max

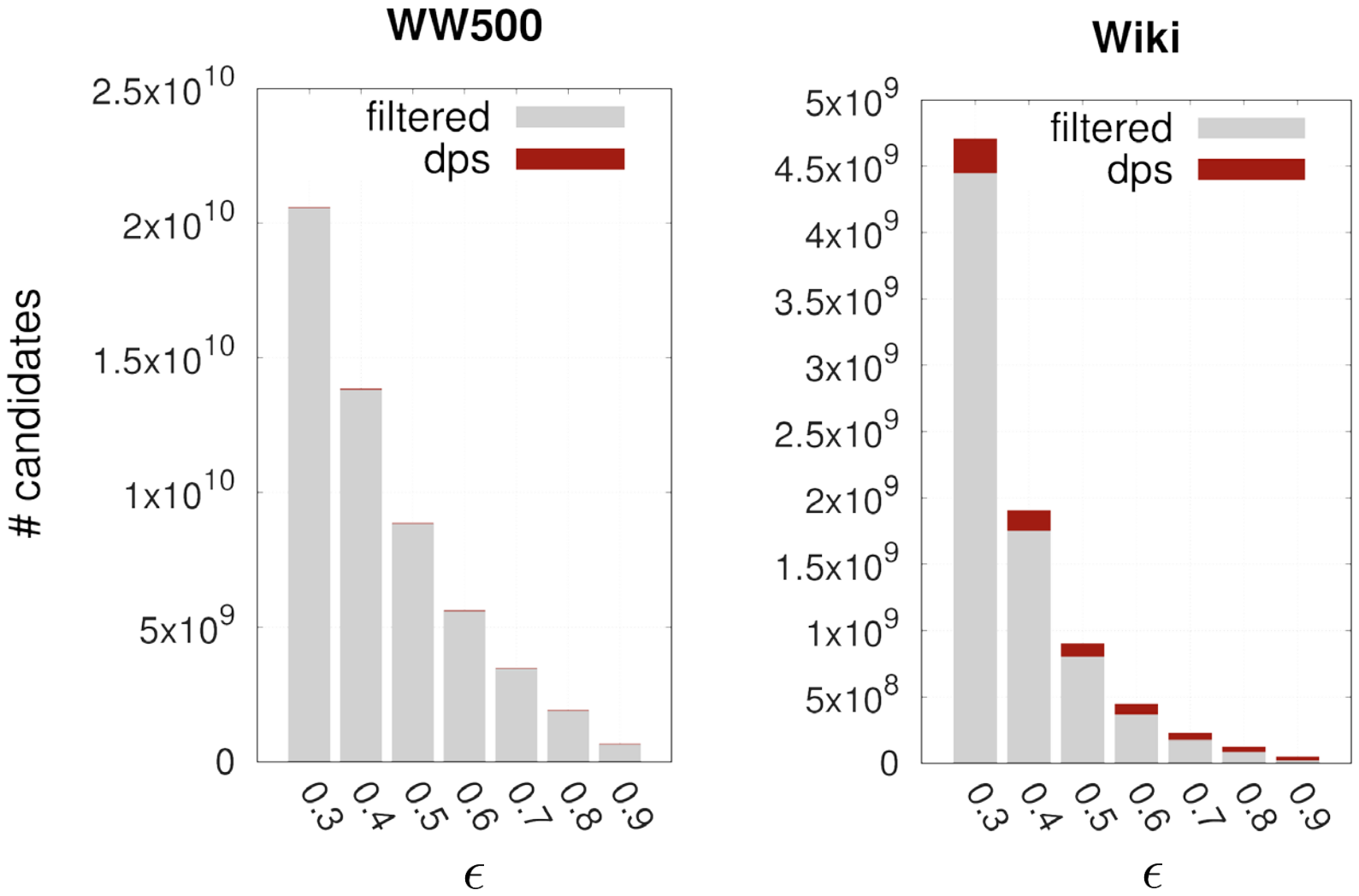
Indexing: in practice

- L2AP indexes fewer non-zeros than previous approaches
- Leads to greatly improved execution runtime



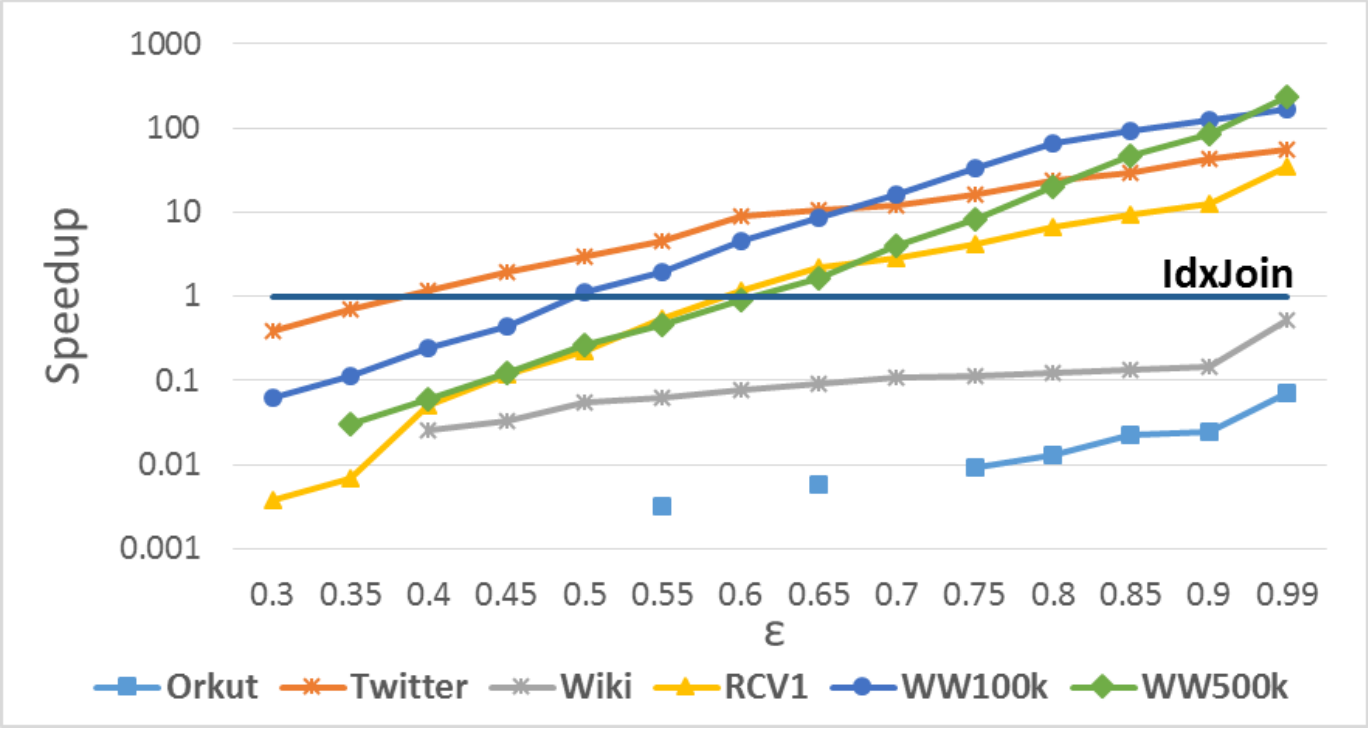
Pruning: in practice

- L2AP filters most objects without computing their similarity



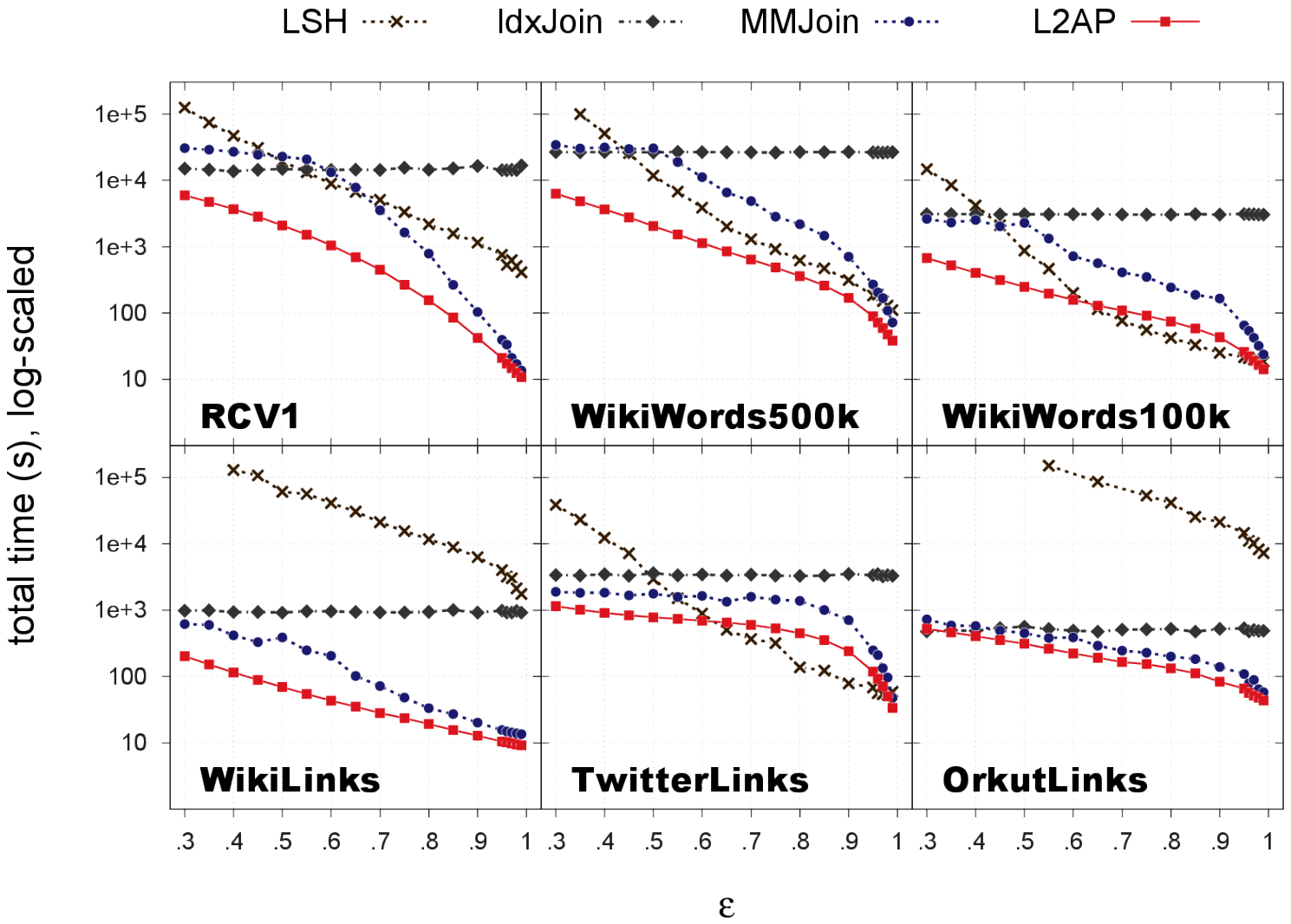
LSH vs. IndexJoin

- In all experiments, LSH parameters were tuned to achieve at least 95% accuracy.
- LSH outperforms IndexJoin at high thresholds.
- Performs poorly at low thresholds and for high dimensional datasets (Orkut, Wiki).



L2AP results: efficiency comparison

- L2AP outperforms all exact and most approximate baselines



Outline

- Nearest neighbor (NN) search
  - IdxJoin – a straightforward solution
  - TAPNN – Tanimoto All-Pairs similarity search
  - L2AP – Cosine All-Pairs similarity search
  - pL2AP – Parallel All-Pairs similarity search
  - Distributed similarity graph construction
- Ongoing and future work

pL2AP

- Shared memory parallel extension of L2AP
- Parallelizing each iteration of L2AP (finding neighbors for one object)
  - has limited potential for work sharing
- Main challenge: avoid synchronization and resource contention

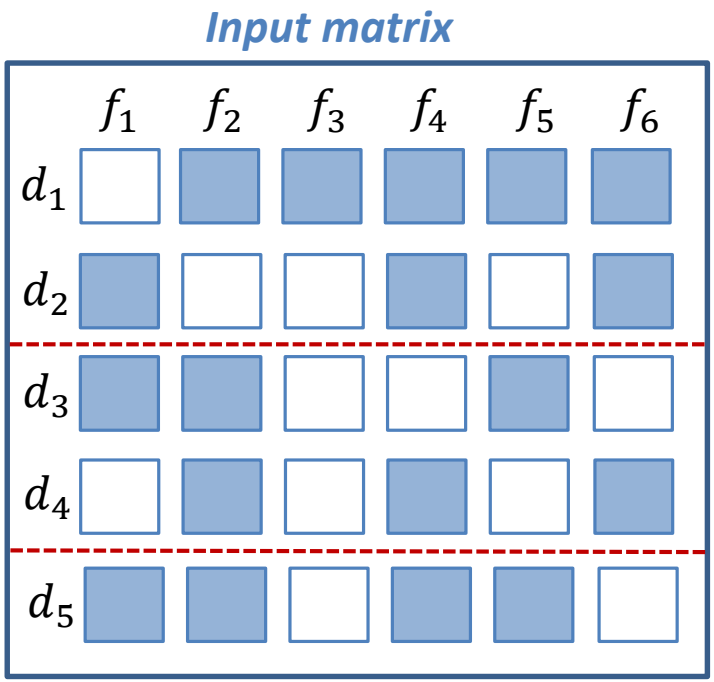
pL2AP

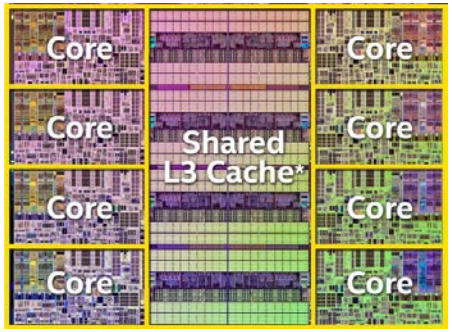
Strategy:

- Precompute the partial inverted index for all objects
- Share the index among the threads
- Use tiling to improve cache locality
  - Index divided into blocks based on non-zeros
  - Threads cooperate to process subset of queries on one index block at a time
- Reduce size of query object data structures
  - Mask-based hash table

pL2AP: Tiling

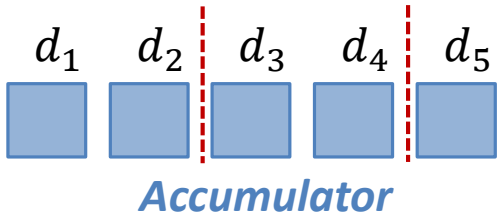
- Split based on index non-zeros & # rows
- During candidate generation, fit index & accumulator in cache





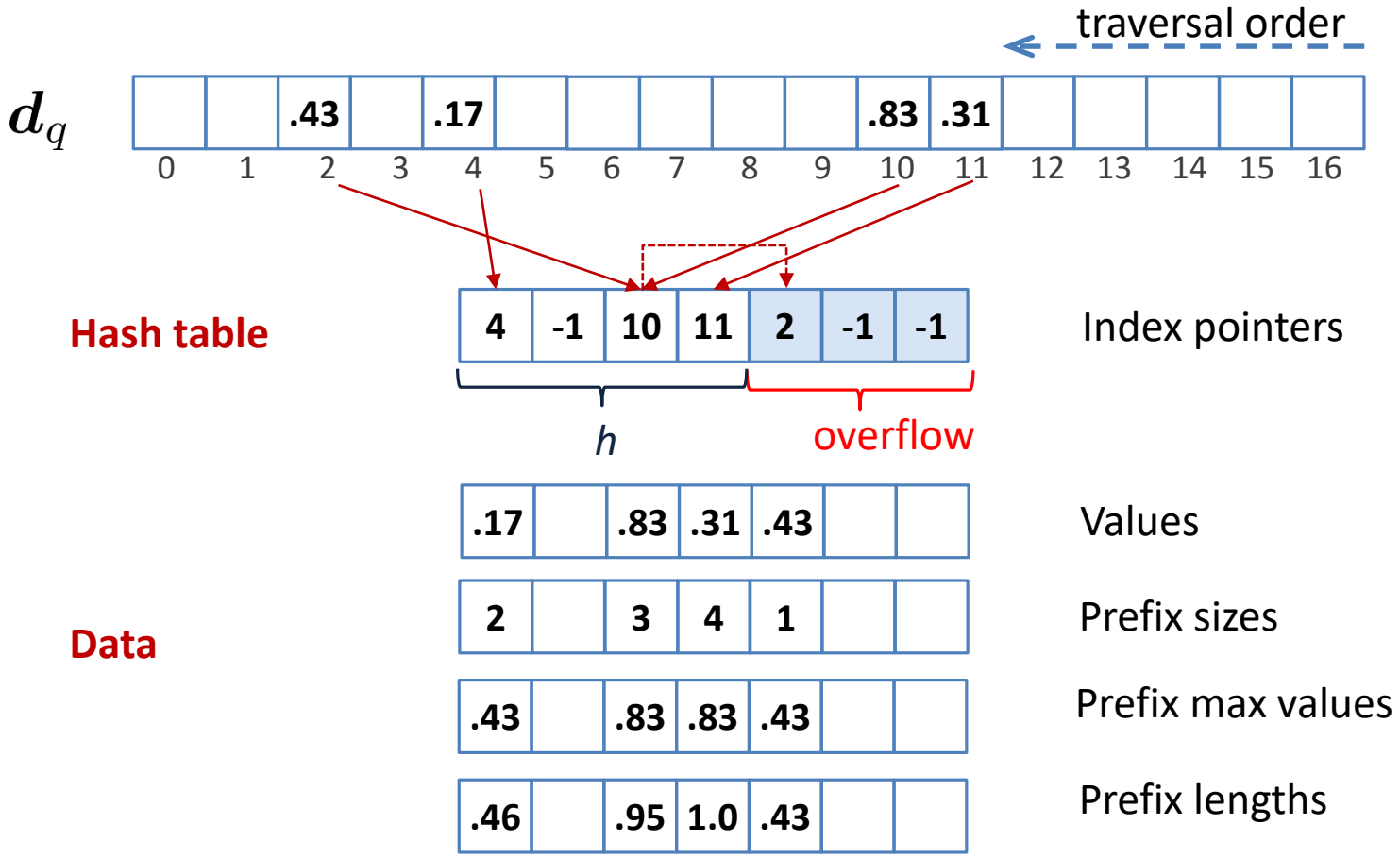
Intel Haswell-E chip

From: http://www.extremetech.com/wp-content/uploads/2014/09/Haswell-Labeled.jpg



pL2AP: Masked hash table

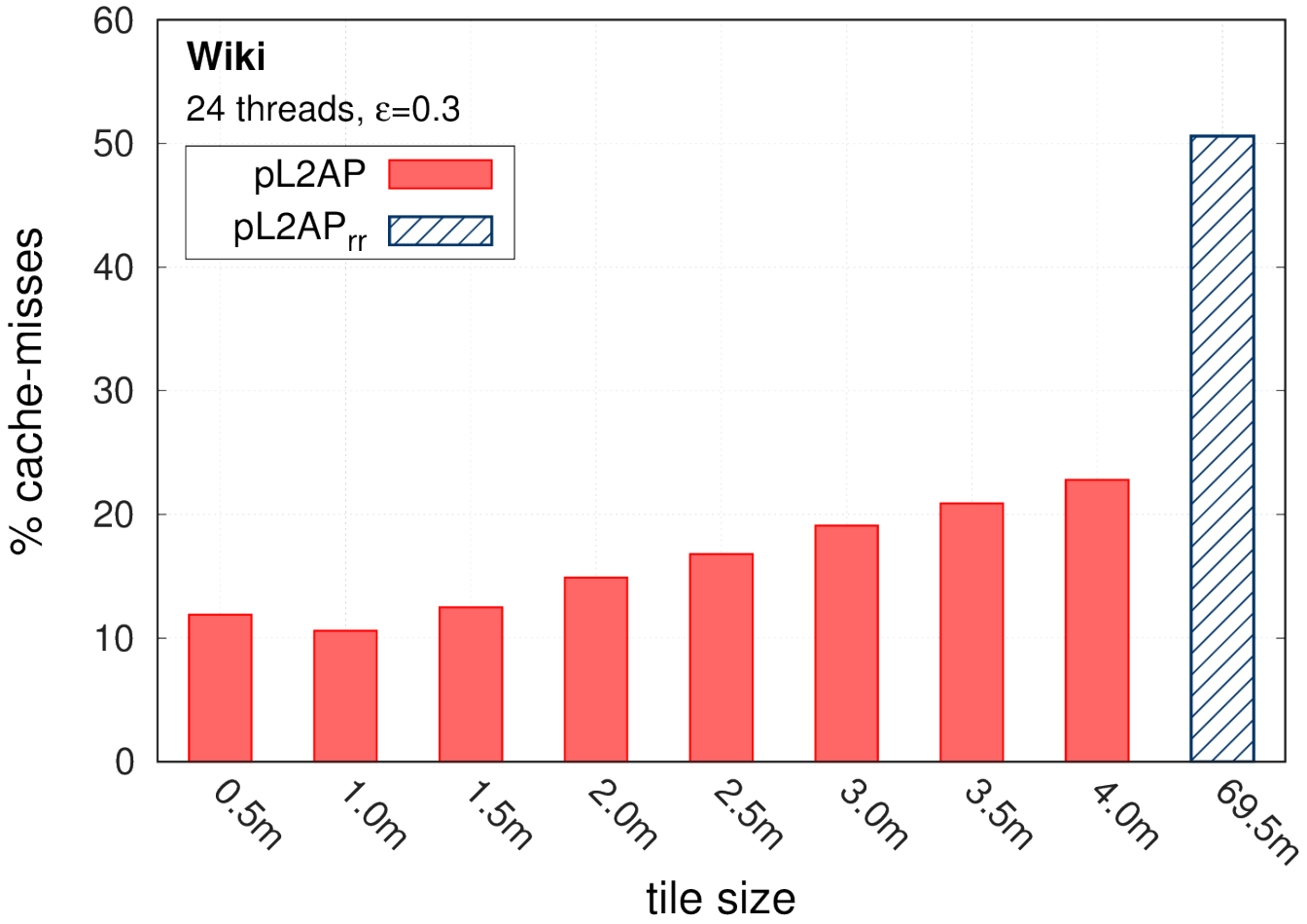
- Fit query vector in cache



Partial linear overflow scan during collision lookup.

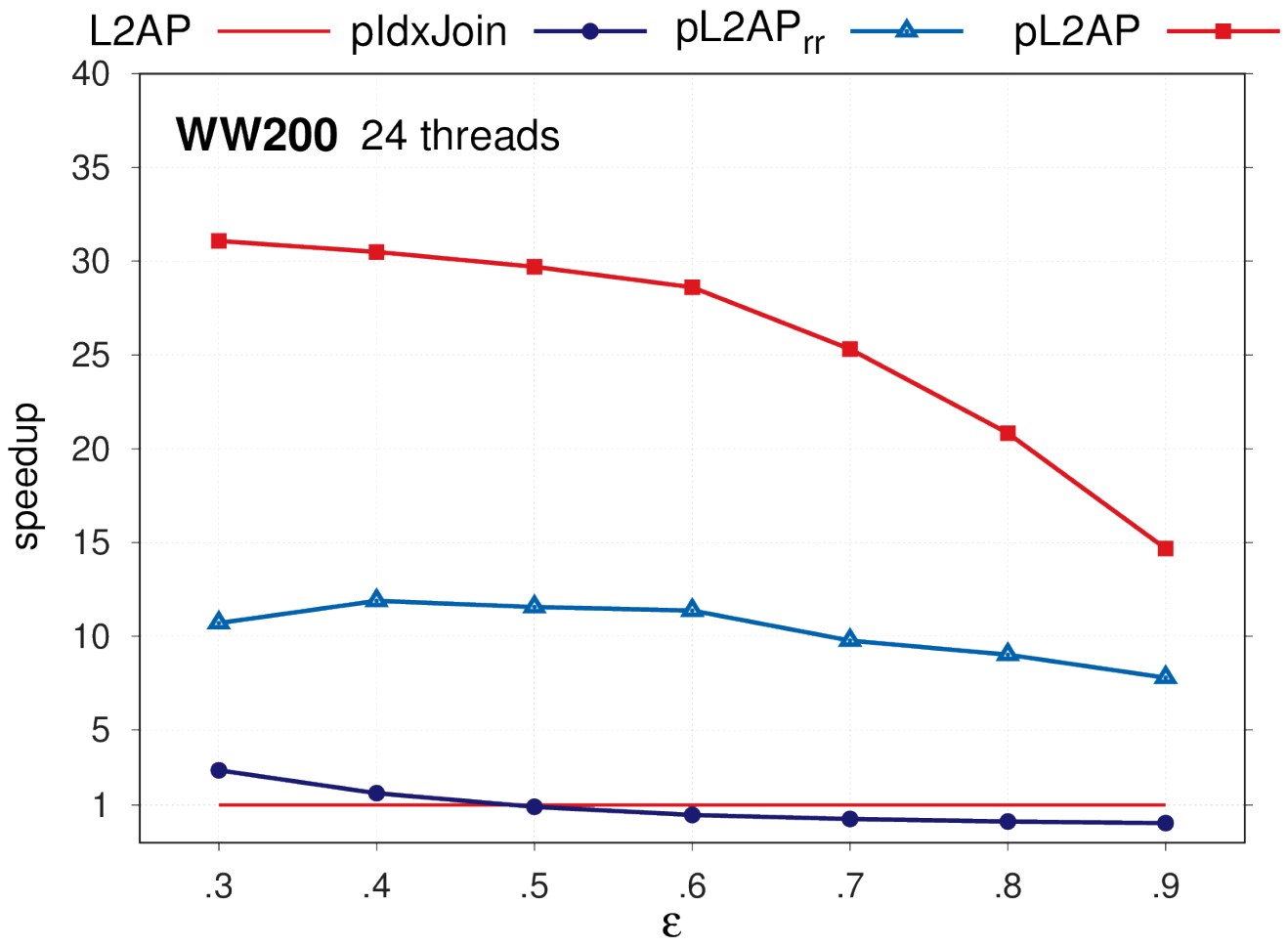
pL2AP: in practice

- Tiling and masked hash table reduce cache misses by more than 50%



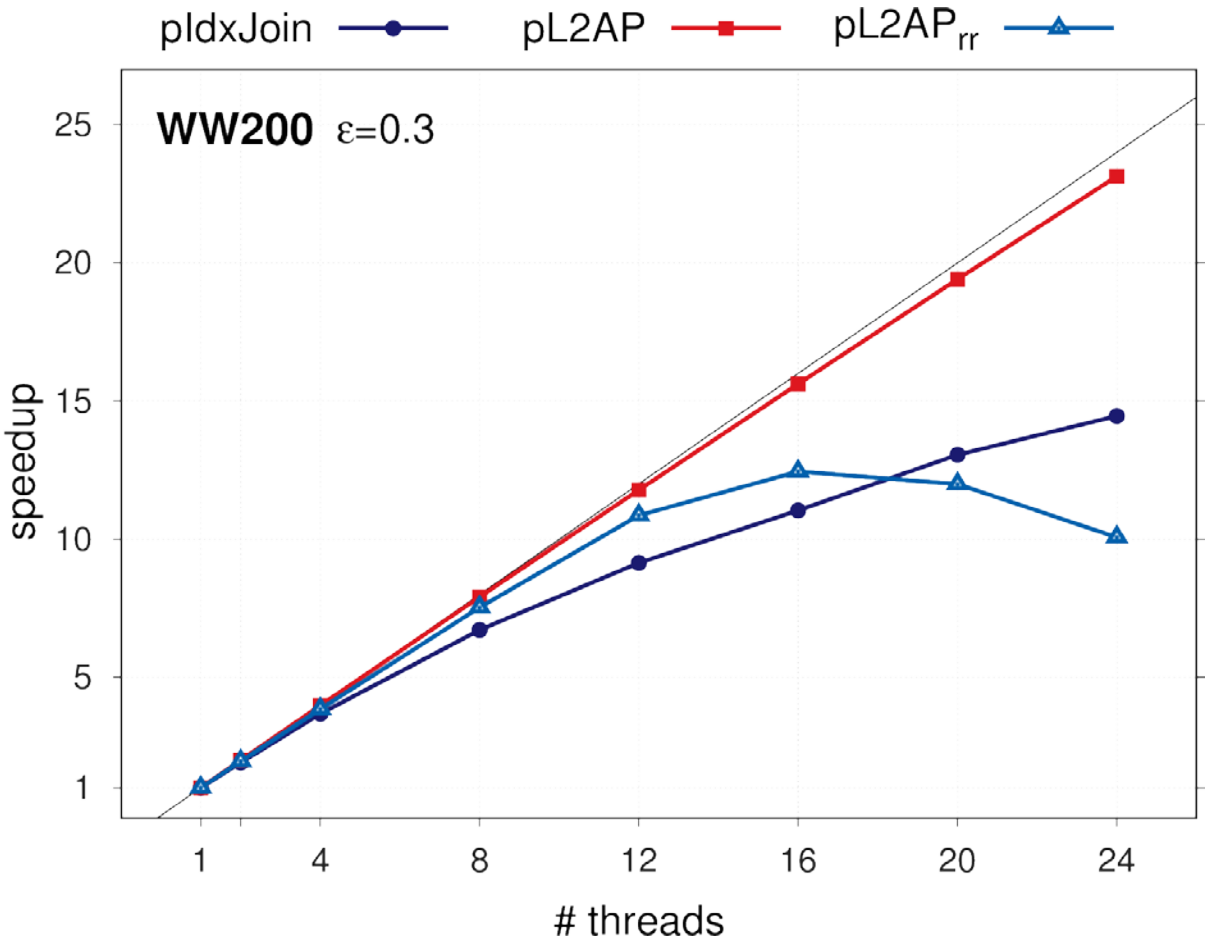
pL2AP: in practice

- pL2AP significantly outperforms parallel baselines, especially at lower values



pL2AP: in practice

- pL2AP shows very good strong scaling, better than that of baselines, especially at lower values



Outline

- Nearest neighbor (NN) search
  - IdxJoin – a straightforward solution
  - TAPNN – Tanimoto All-Pairs similarity search
  - L2AP – Cosine All-Pairs similarity search
  - pL2AP – Parallel All-Pairs similarity search
  - Distributed similarity graph construction
- Ongoing and future work

Distributed memory parallel algorithm

- Assume input partitioned among *p* different machines
- Explore ways to minimize communication cost specific to the problem

Distributed memory parallel algorithm

- Currently exploring:

  Dynamic candidate generation and verification assignments

  - For high thresholds, inverted index very small
  - Compare blocks B_i and B_j
  - Send inverted index of B_i to B_j
  - B_j finds candidates for all queries in B_j
  - Decide what is better: send partial similarities to B_i or get partial forward index from B_i?

Distributed memory parallel algorithm

- Currently exploring:

  Blocking strategies that can eliminate some object comparisons, based on:

  - Holder inequalities

$$|\langle \boldsymbol{d}_i, \boldsymbol{d}_j \rangle| \leq ||\boldsymbol{d}_i||_1 \times ||\boldsymbol{d}_j||_\infty$$
$$|\langle \boldsymbol{d}_i, \boldsymbol{d}_j \rangle| \leq ||\boldsymbol{d}_i||_\infty \times ||\boldsymbol{d}_j||_1$$

  - Euclidean distance of normalized vectors

$$\mathrm{C}(\boldsymbol{d}_i, \boldsymbol{d}_j) = 1 - \frac{1}{2}\mathrm{E}^2(\hat{\boldsymbol{d}}_i, \hat{\boldsymbol{d}}_j)$$

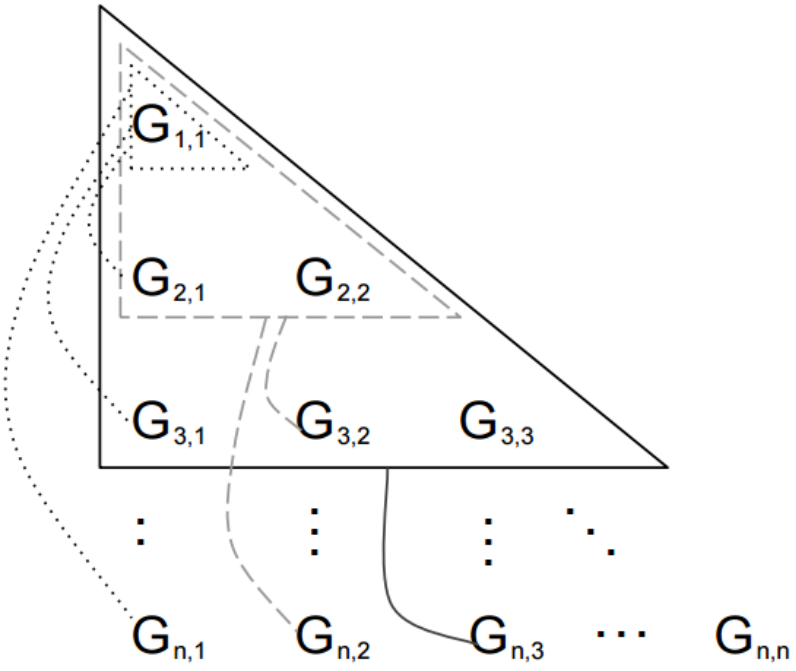


Image from Optimizing Parallel Algorithms for All Pairs Similarity Search, by Maha Alabduljalil et al.

Distributed memory parallel algorithm

- Currently exploring:

  Graph partitioning based load estimate

  – Over-partition blocks
  – Model block-block nn search based on nnzs and number of objects in candidate blocks
  – Use graph partitioning to assign blocks to processes

Outline

- Nearest neighbor (NN) search
  - IdxJoin – a straightforward solution
  - TAPNN – Tanimoto All-Pairs similarity search
  - L2AP – Cosine All-Pairs similarity search
  - pL2AP – Parallel All-Pairs similarity search
  - Distributed similarity graph construction
- Ongoing and future work

Nearest neighbors directions

Ongoing work:

- Distributed neighbor search (MPI+OMP)
  - Challenge: minimize communication + load balance
  - Solution: graph partitioning based load estimate, staged communication (send partial inverted index + what is needed of forward index)
- *SnnLib*, an integrated Sparse NN library
  - Written in C, with C++, Python, Matlab, R bindings
  - Include both self and non-self joins
  - Include shared memory and distributed code

L2AP: http://davidanastasiu.net/software/l2ap/

L2Knng: http://davidanastasiu.net/software/l2knng/

Future directions: short term

Nearest Neighbors:

- Beyond cosine similarity
  - Preliminary theoretic results for Dice similarity

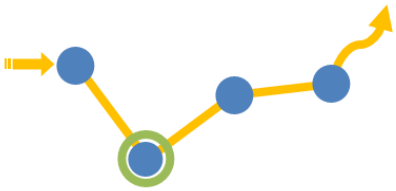
$$\mathrm{D}(d_i, d_j) = 2\frac{\left\langle \mathbf{d}_i, \mathbf{d}_j \right\rangle}{||\mathbf{d}_i||_2^2 + ||\mathbf{d}_j||_2^2} \qquad T(\boldsymbol{d}_i, \boldsymbol{d}_j) \leq D(\boldsymbol{d}_i, \boldsymbol{d}_j) \leq C(\boldsymbol{d}_i, \boldsymbol{d}_j)$$

  - Inner-product similarity
  - Euclidean distance
- Beyond OMP+MPI
  - GPGPU
  - Xeon Phi
  - Heterogeneous clusters, Hadoop, Spark

Research interests

Behavior-centric analytics

- Understand human behavior
  - Characterizing micro-behavior evolution
  - Integrate heterogeneous behavior signals



Geolocation

Click-stream

Internet-of-things/mobile sensors

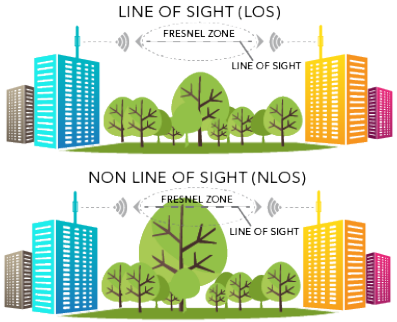
Social networks

Online learning

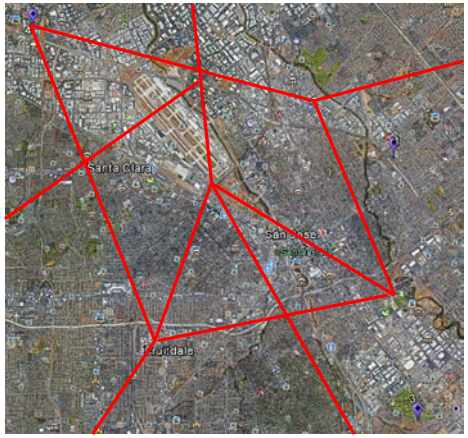
Research interests

Wireless Emergency Network Topology

- Goal: Identify optimal placement of antennae to create a strong and redundant wireless emergency network for the City of San Jose
- Solution: two steps
  - Efficient line-of-site probability estimation
  - Path-constrained multi-criteria optimization using heterogeneous data



- Criteria
  - Minimum LOS links
  - Must-have locations
  - Preferred locations
  - Hardware limitations (P2P vs. sector antennae)
  - Signal strength
- Data
  - Satellite data (SRTM/NED)
  - OpenStreetMaps
  - Mapzen San Jose
  - Survey markers
  - City LIDAR topography
  - Census
  - WEN data & preferences



Research interests

What are your interests?

Questions?

References

[1] David C. Anastasiu and George Karypis. L2AP: Fast Cosine Similarity Search With Prefix L-2 Norm Bounds. Proceedings of the 30th IEEE International Conference on Data Engineering (ICDE 2014).
[2] Dongjoo Lee, Jaehui Park, Junho Shim, and Sang-goo Lee. 2010. An efficient similarity join algorithm with cosine similarity predicate. In Proceedings of the 21st international conference on Database and expert systems applications: Part II (DEXA'10), Pablo Garcia Bringas, Abdelkader Hameurlain, and Gerald Quirchmayr (Eds.). Springer-Verlag, Berlin, Heidelberg, 422-436
[3] M. Kryszkiewicz. Bounds on lengths of real valued vectors similar with regard to the tanimoto similarity. Intelligent Information and Database Systems, ser. Lecture Notes in Computer Science, A. Selamat, N. Nguyen, and H. Haron, Eds. Springer Berlin Heidelberg, 2013, vol. 7802, pp. 445-454.
[4] ----. Using non-zero dimensions for the cosine and tanimoto similarity search among real valued vectors. Fundamenta Informaticae, vol. 127, no. 1-4, pp. 307-323, 2013.
[5] ----. Using non-zero dimensions and lengths of vectors for the tanimoto similarity search among real valued vectors. Intelligent Information and Database Systems. Springer International Publishing, 2014, pp. 173-182.