Efficient Large Scale Maximum Inner Product Search

Presenter: BAO Ergute
Supervisor: James CHENG

Table of Contents

Problem Definition

Maximum Inner Product Search (MIPS):

Given a collection of "database" vectors $S \subset R^d$ and a query $q \in R^d$, find a data vector maximizing the inner product with the query:

$$p = \mathbf{argmax}_{x \in S} q^T x$$

It is worthy to mention that MIPS is closely related to **Nearest Neighbor Search (NNS)**:

$$p = \mathbf{argmin}_{x \in S} \|q - x\|_2$$

Problem Definition

Maximum Inner product Search v.s. **Nearest Neighbor Search (NNS)**

Nearest Neighbor Search:

$$\|q - x\|_2^2 = \|q\|_2^2 + \|x\|_2^2 - 2q^T x$$

WLOG, we assume $\|q\|_2 = 1$, since we can always normalize it otherwise. Then we have

$$\|q - x\|_2^2 = 1 + \|x\|_2^2 - 2q^T x$$

We can see that MIPS and NNS are equivalent when $\|x\|_2 = c$ (some constant c) for all $x \in S$.

Literature Review

- **Locality Sensitive Hash based methods for MIPS:**

L2-ALSH (Shrivastava and Li)
Simple LSH/ALSH (Neyshabur and Srebro)

Idea: Get rid of $\|x\|_2$ in the dataset.

- **Graph Nearest Neighbor Search based on K Nearest Neighbor Graph**

Fast Approximate Nearest-Neighbor Search with k-Nearest Neighbor Graph (Hajebi et al)

Literature Review

L2-ALSH

1. **Transformation:** For an integer parameter m and real valued parameters $0 < U < 1$, consider the following pair of mappings:

$$P(x) = [Ux; \|Ux\|_2^2; \|Ux\|_2^4; \ldots; \|Ux\|_2^{2^m}]$$

$$Q(q) = [q; 1/2; 1/2; \ldots; 1/2]$$

We obtain:

$$\|P(x)\|_2^2 = \|Ux\|_2^2 + \|Ux\|_2^4 + \ldots + \|Ux\|_2^{2^m} + \|Ux\|_2^{2^{m+1}}$$

$$\|Q(q)\|_2^2 = \|q\|_2^2 + m/4 = 1 + m/4$$

$$\|Q(q) - P(x)\|_2^2 = (1 + m/4) - 2Uq^T x + \|Ux\|_2^{2^{m+1}}$$

Note: $\|Ux\|_2^{2^{m+1}} \to 0, m/4 \to 0$ as $m \to \infty$, also U is a constant.

Now we have: $\mathbf{argmax}_{x \in S} q^T x = \mathbf{argmin}_{x \in S} \|Q(q) - P(x)\|_2$ as $m \to \infty$.

Literature Review

L2-ALSH

Now that we have the equivalence between MIPS and NNS, we can use LSH to do NNS

2. LSH

$$h_{a,b}^{L_2}(x) = \lfloor \frac{a^T x + b}{r} \rfloor$$

Summary:

0. Scale x and normalize q.

   $R^d \rightarrow R^d$, the relative order of $q^T x$ is preserved

1. Asymmetric transformations on scaled x and normalized q.

   $R^d \rightarrow R^{d+m}$, MIPS to NNS

2. LSH for Euclidean distance

Discussion: $m \rightarrow \infty$ is impossible.

Literature Review

Simple-LSH

1. **Transformation**: consider the following mapping for both x and q:

$$P(x) = [x; \sqrt{1 - \|x\|_2^2}]$$

with a normalized query q and scaled data x

$$P(q) = [q; 0]$$

We obtain:

$$P(q)^T P(x) = q^T x$$

Now we have: $\mathbf{argmax}_{x \in S} q^T x = \mathbf{argmax}_{x \in S} P(q)^T P(x)$.

Note: $\|P(x)\|_2 = \|P(q)\|_2 = 1$.

2. **LSH**

$h_a(x) = \mathbf{sign}(a^T x)$

Literature Review

Simple-LSH

If $q^T x \geq S$, then

$$Pr[h_a(P(q)) = h_a(P(x))] \geq 1 - \frac{\cos^{-1}(S)}{\pi}$$

If $q^T x \leq cS$, then

$$Pr[h_a(P(q)) = h_a(P(x))] \leq 1 - \frac{\cos^{-1}(cS)}{\pi}$$ (Goemans and Wiliamson)

Since for any $0 \leq x \leq 1$, $1 - \frac{\cos^{-1}(x)}{\pi}$ is a monotically increasing function, this gives us an LSH.

Summary:

0. Scale x and normalize q.
1. Transformation on scaled x and normalized q.
   $R^d \rightarrow R^{d+1}$, the value of $q^T x$ is preserved.
2. LSH for angluar distance

Literature Review

Simple-ALSH (q not normalized)

1. **Transformation:** consider the following asymmetric mappings for x and q:

$$P(x) = [x; \sqrt{1 - \|x\|_2^2}; 0]$$

$$Q(q) = [q; 0; \sqrt{1 - \|q\|_2^2}]$$

with a scaled query q and data x

We obtain:

$$Q(q)^T P(x) = q^T x$$

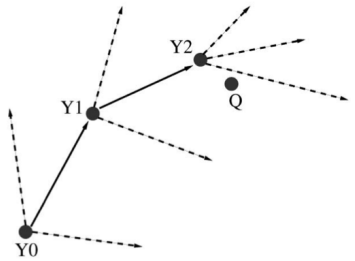
- Provides a LSH based method for normalized/un-normalized query q
- Easy to implement (append digits)
- Enjoys the same theoretical guarantees as LSH

Literature Review

Graph Nearest Neighbor Search based on K Nearest Neighbor Graph

Intuition: A neighbor of my neighbor is likely to be my neighbor.

At each iteration, the algorithms picks the nearest node w.r.t the query to expand.



The time complexity is bounded for L1 norm. Complexity of L2 norm is not guaranteed.

Pros: Very good empirical performance

Cons: No theoretical guarantee. Construction is time consuming.

Generalization from K-NN Graph to K-MIP Graph: Both inner product and L2 distance are metrics for measuring the similarity. We can use the same idea in Graph Nearest Neighbor Search to do Maximum Inner Product Search.

K-MIP Graph for Maximum Inner Product Search

Expand according to $\frac{p^T q}{\|p\|_2 \|q\|_2}$

Reason for using $\frac{p^T q}{\|p\|_2 \|q\|_2}$: if $p = cq$, for some constant $c > 0$. Then p and q has the same MIP neighbors. Yet this heuristic results in poor performance.

Algorithm:

Prerequisite: Graph G, a K-MIP graph for the dataset D.

- $S = \{\}$ # set of MIP candidates
- Pick a start node x from D
- For $i = 1,2,...,h$ do
  - $y = \text{argmax}_{y \in N(x,G)} \frac{y^T q}{\|y\|_2 \|q\|_2}$ # choose the node with the closest angle to expand
  - $S = S \cup N(y, G)$ # update the set of candidates
  - $x = y$
- end for
- Sort S according to the inner product w.r.t. q, and return the top k elements

K-MIP Graph for Maximum Inner Product Search

Expand according to $p^T q$

Reason for using $p^T q$: find the biggest $p^T q$ matches with our goal.

Use C to keep the candidates to be expanded

Algorithm:

Prerequisite: Graph G, a K-MIP graph for the dataset D.

- $S = \{\}$ # set of MIP candidates
- $C = \{\}$ # set of to be expanded candidates
- Pick a start node x from D # both random initialization or LSH can be used
- For $i = 1,2,...,h$ do
  - $y = \text{argmax}_{y \in C} y^T q$ # choose the node with the biggest inner product from C to expand
  - $S = S \cup N(y, G)$ # update the set of MIP candidates
  - $C = C \cup N(y, G) - y$ # update the set of candidates to be expanded
- end for
- Sort S according to the inner product w.r.t. q, and return the top k elements

Experiments

Dataset information and experiment setup

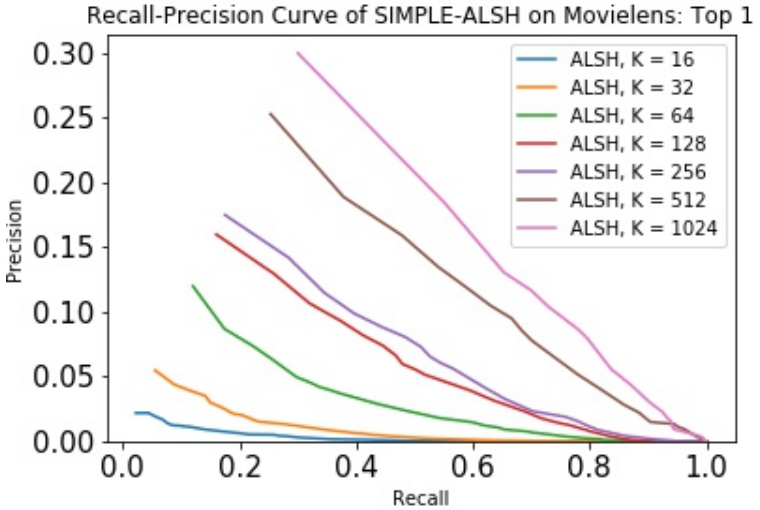
Datasets: Movielens 10M and Netflix (user-item rating matrix M)

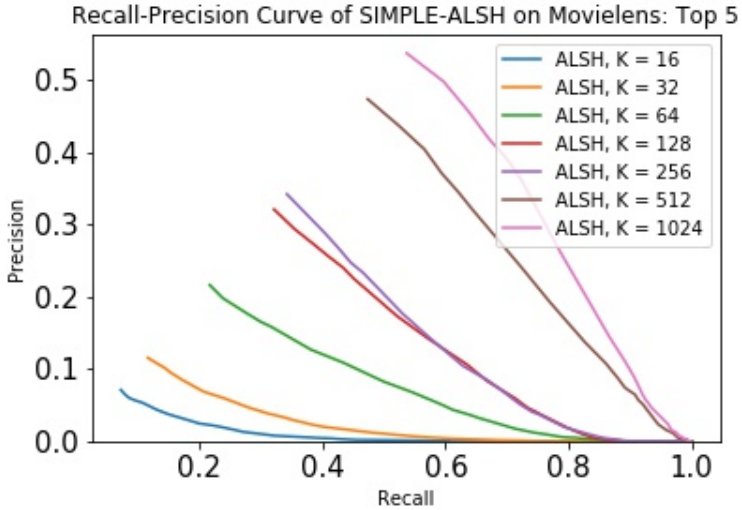
We apply pureSVD to the rating matrix $M = U\Sigma V$. And we set $Q = U\Sigma$ and $S = V$ to be the queries and dataset respectively.

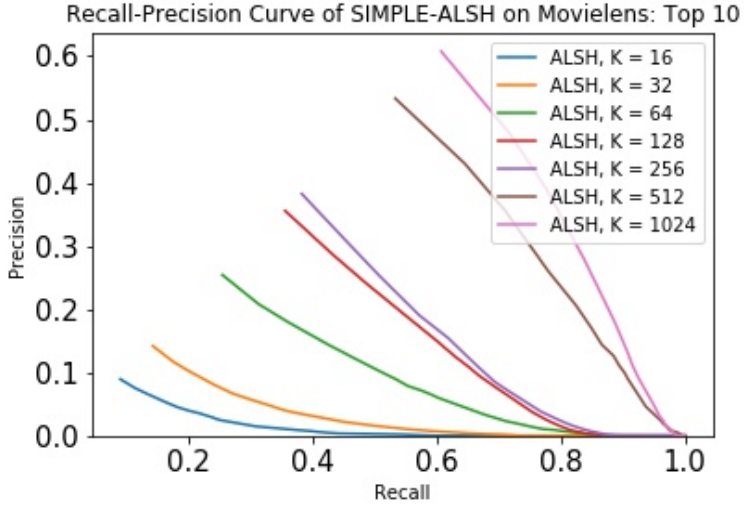
We pick $q \in Q$ to find $p \in S$ s.t. $p^T q$ is among the top k maximum inner products w.r.t q. Here we pick $k \in \{1,5,10\}$.

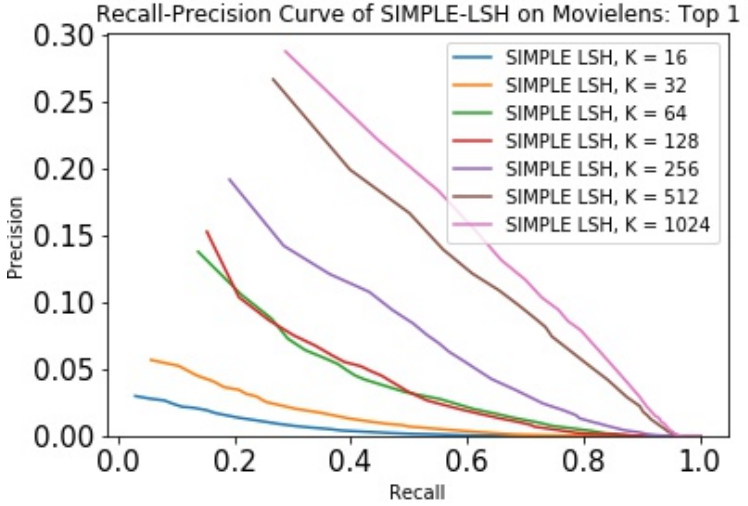
Experiments

Simple LSH/ALSH



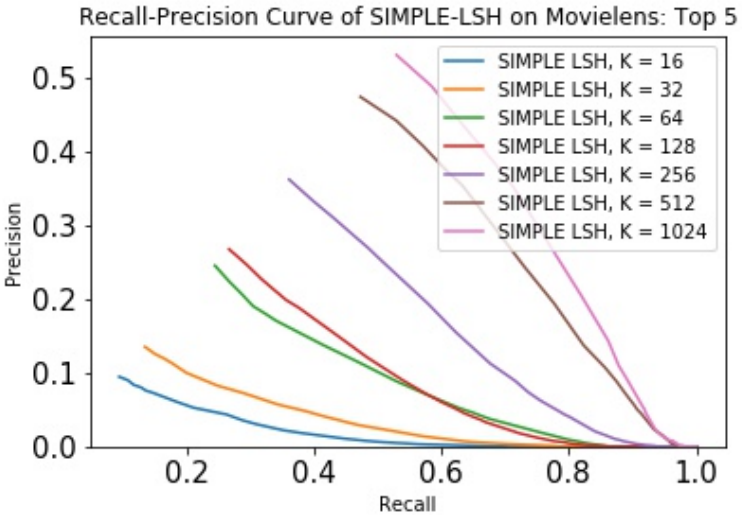


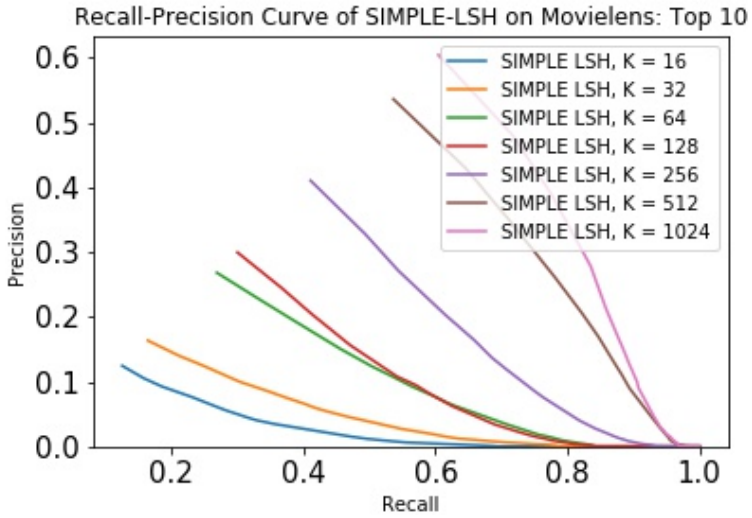




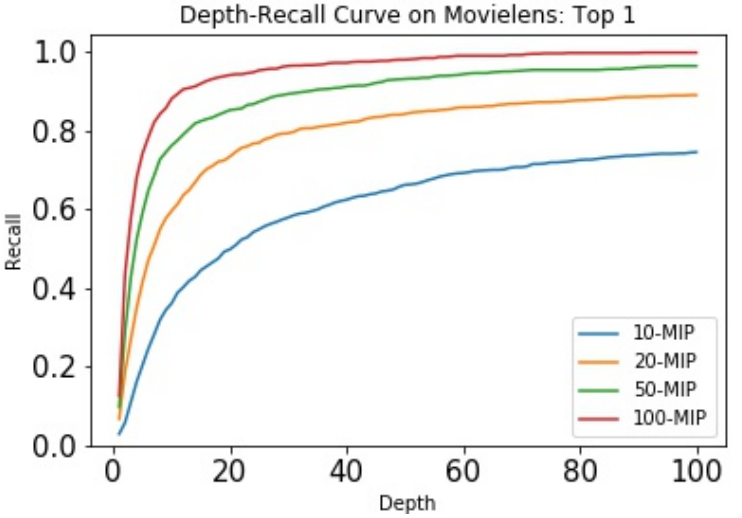
Experiments

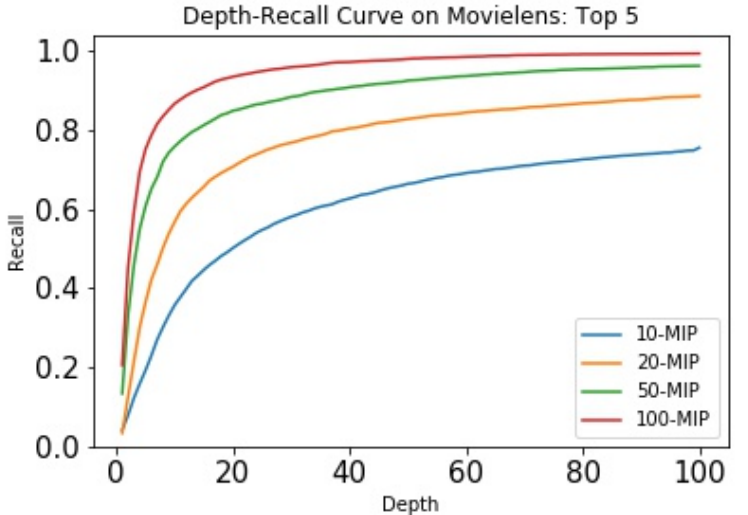
Simple LSH/ALSH





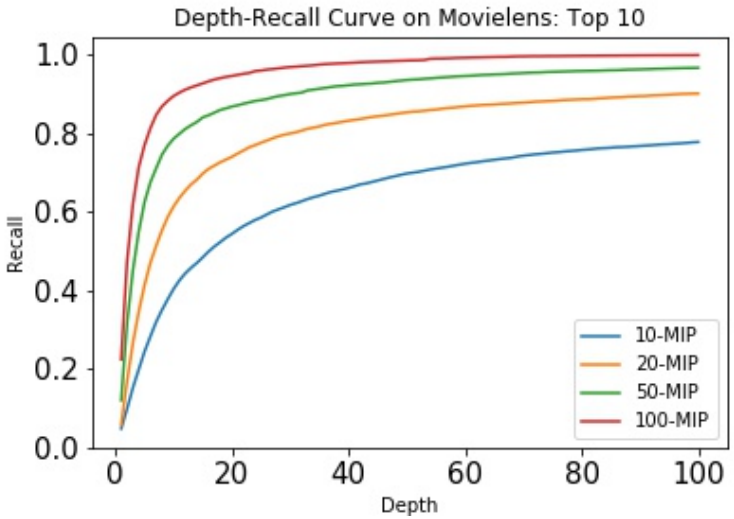
K-MIP Graph

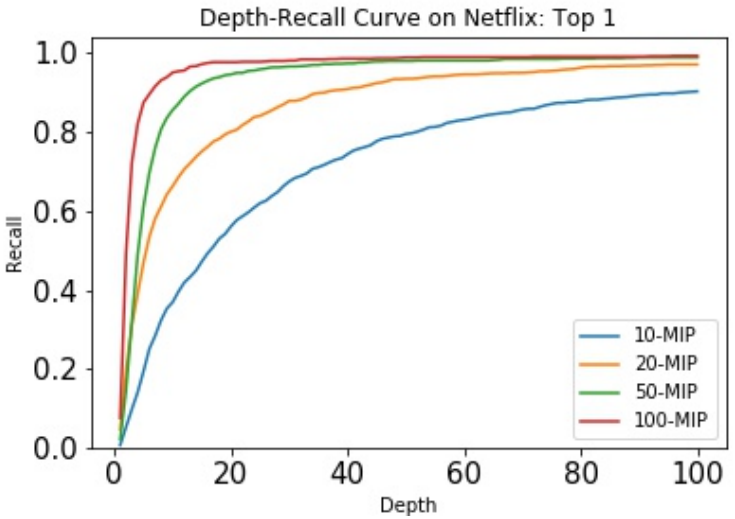


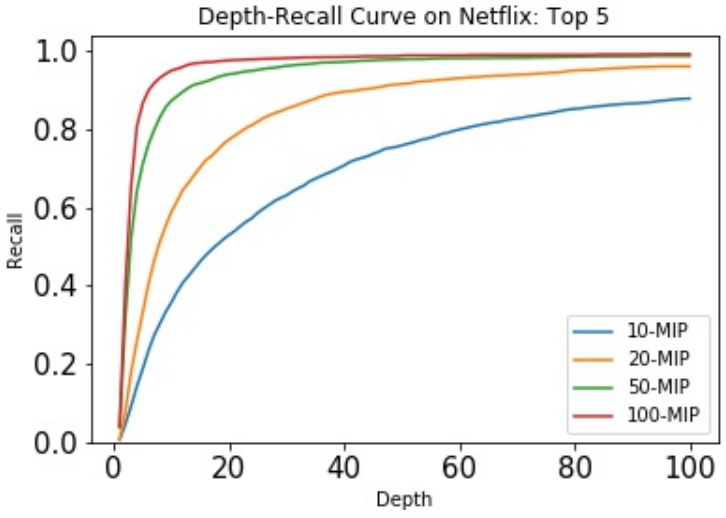


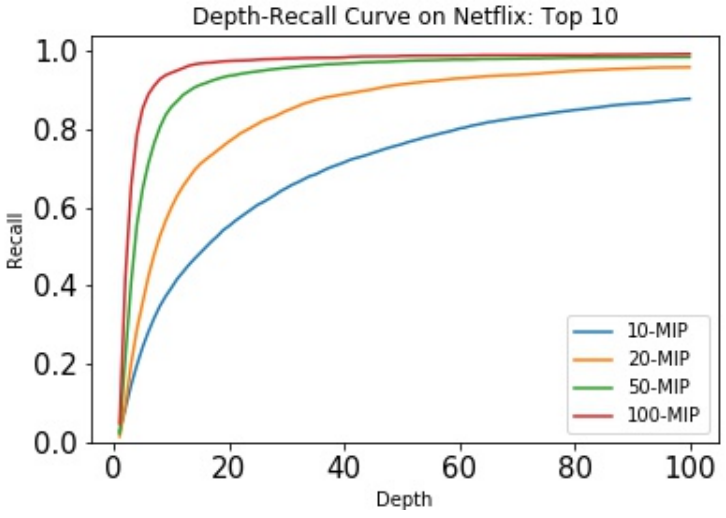
Experiments

K-MIP Graph



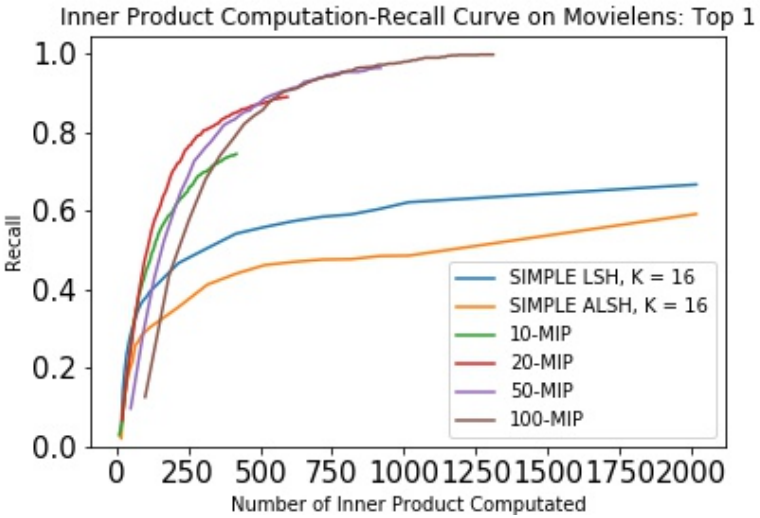


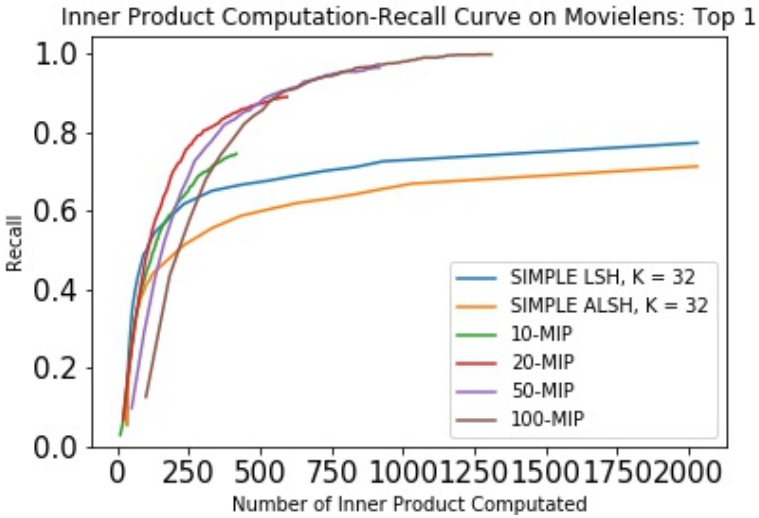


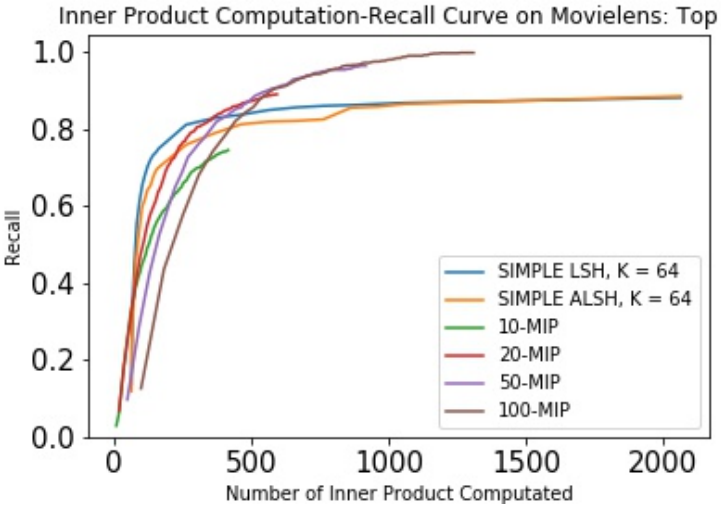


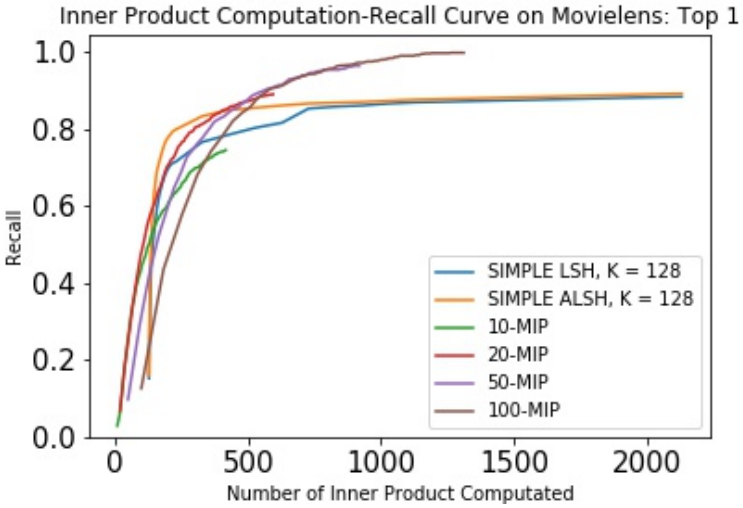
Experiments

K-MIP Graph compared with Simple LSH/ALSH



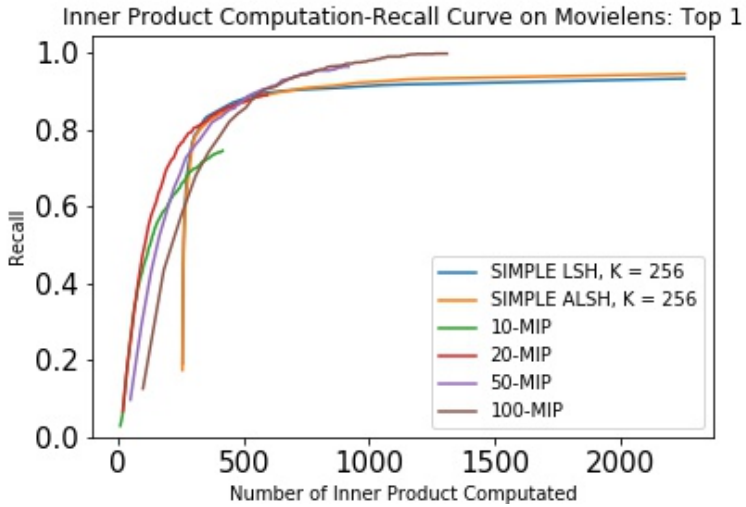


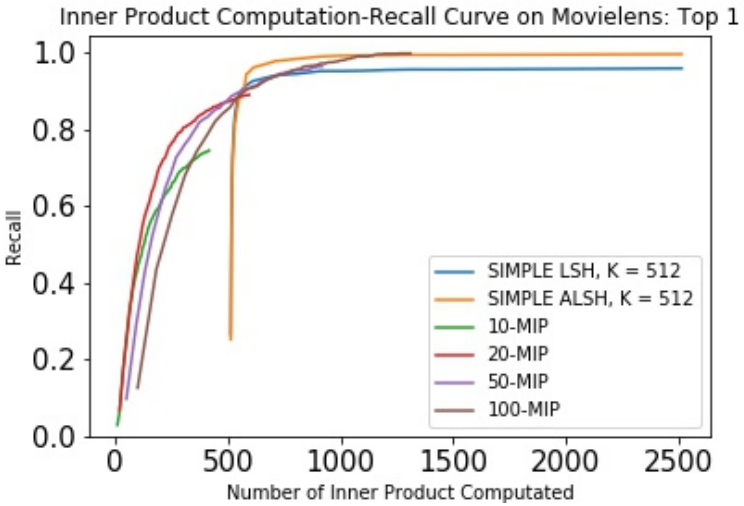


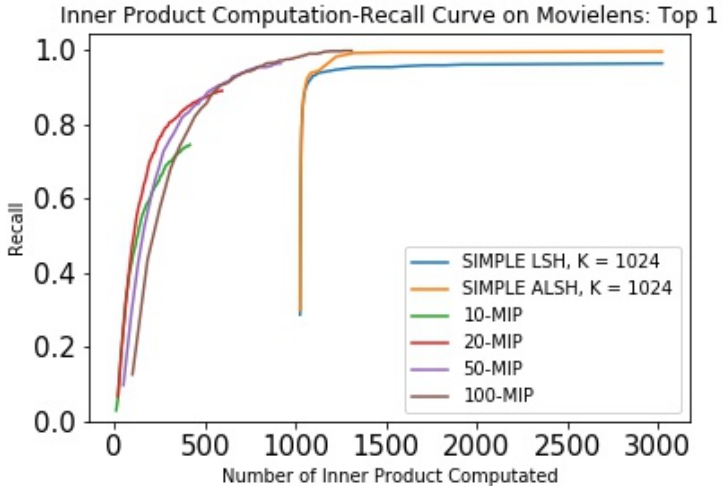


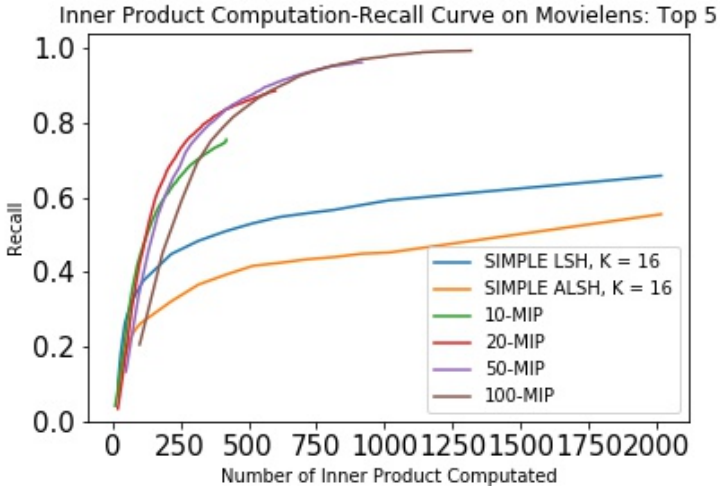
Experiments

K-MIP Graph compared with Simple LSH/ALSH



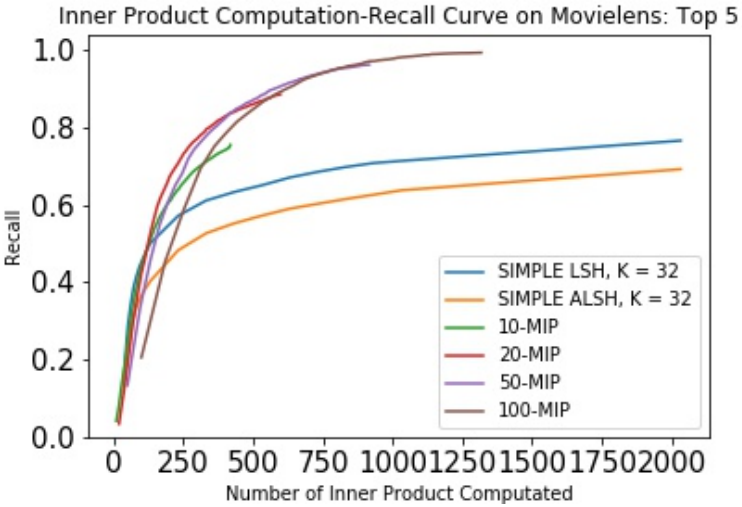


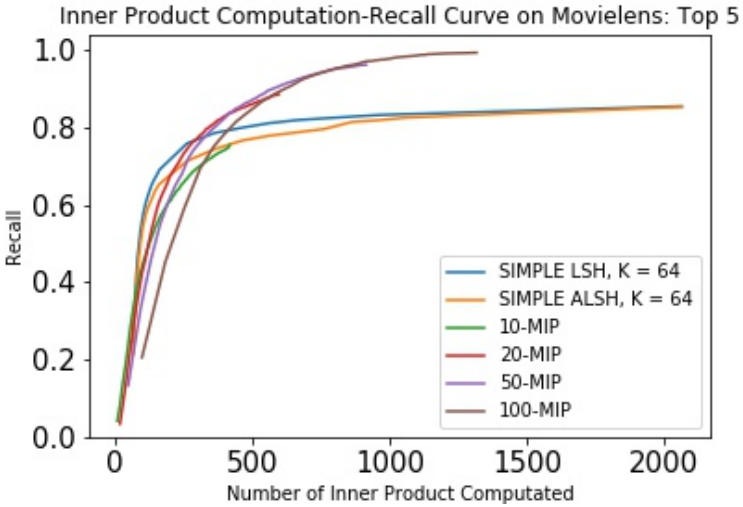


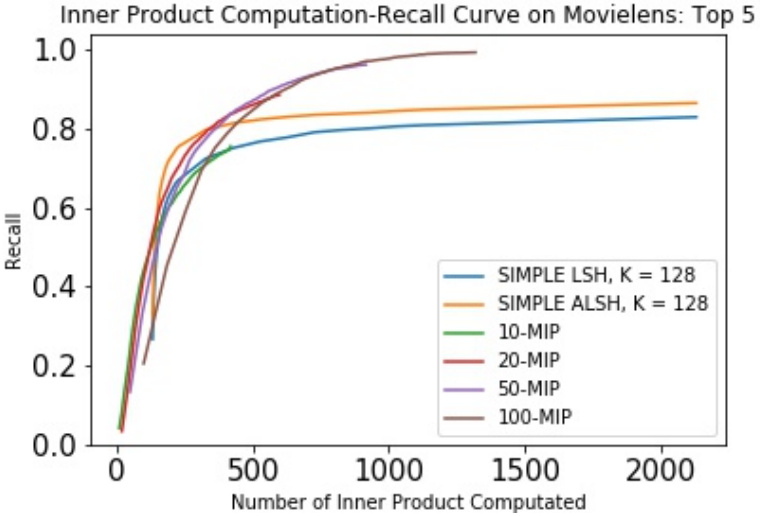


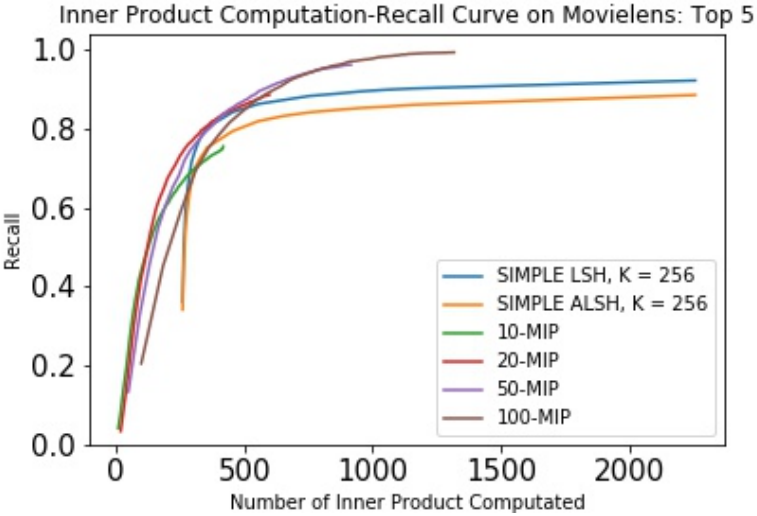
Experiments

K-MIP Graph compared with Simple LSH/ALSH



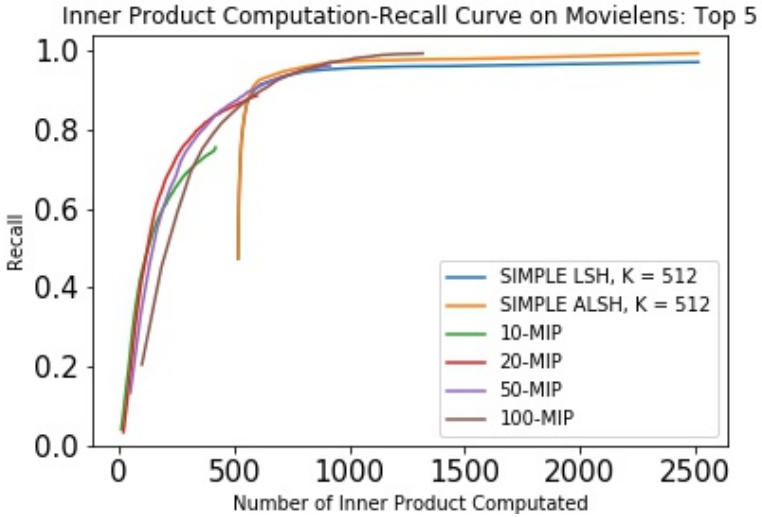


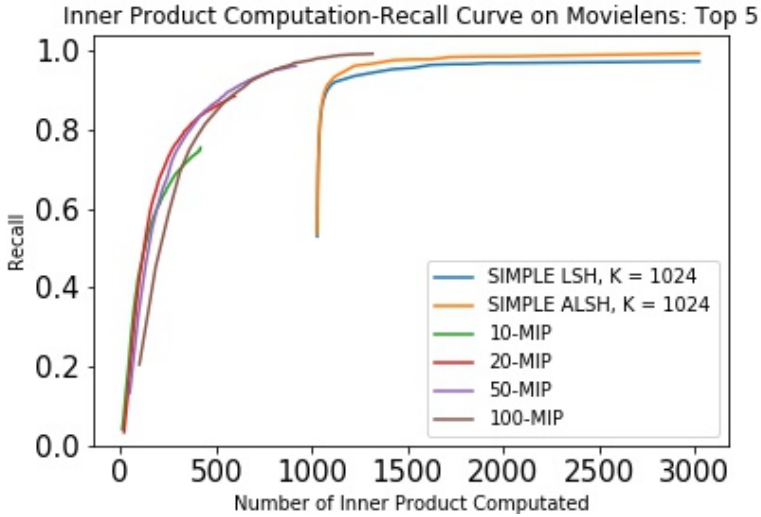


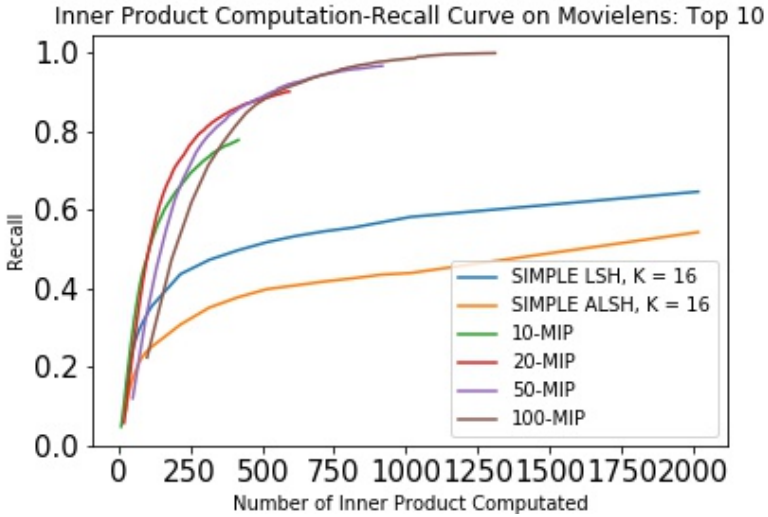


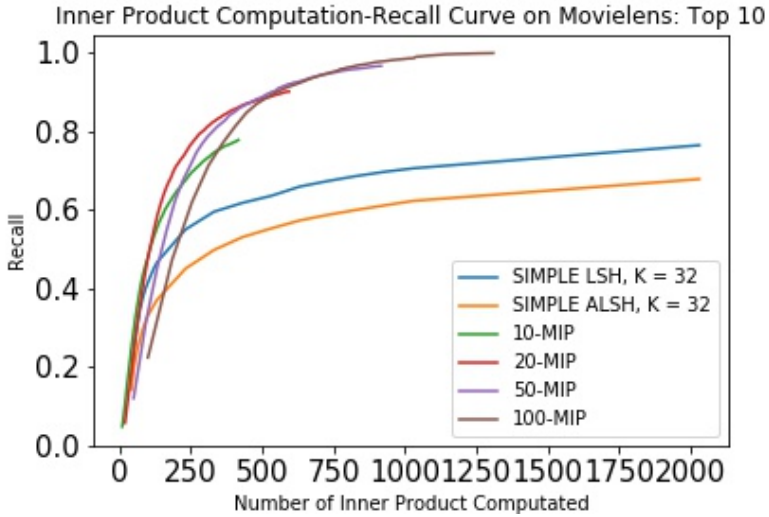
Experiments

K-MIP Graph compared with Simple LSH/ALSH



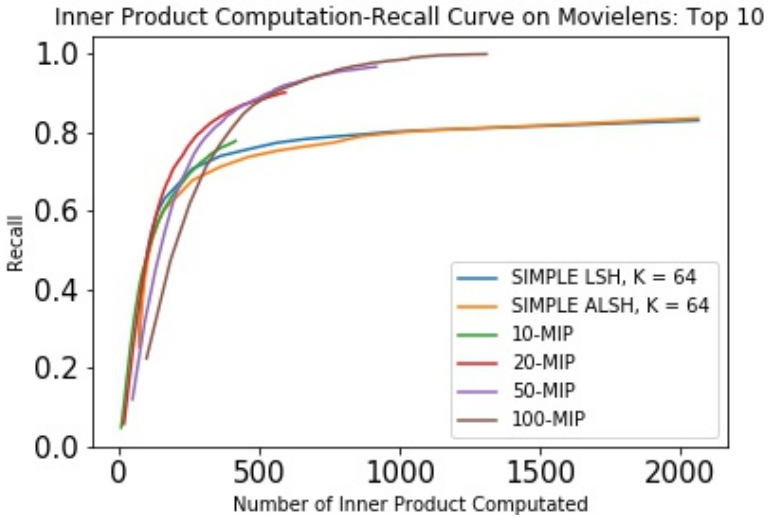


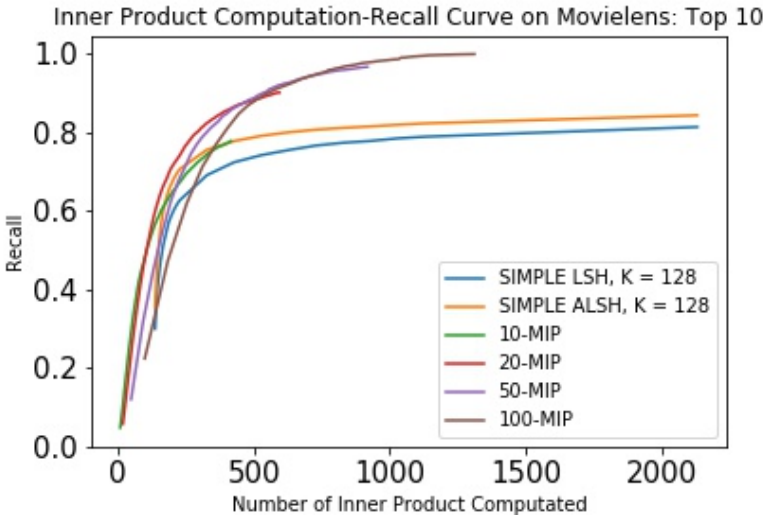


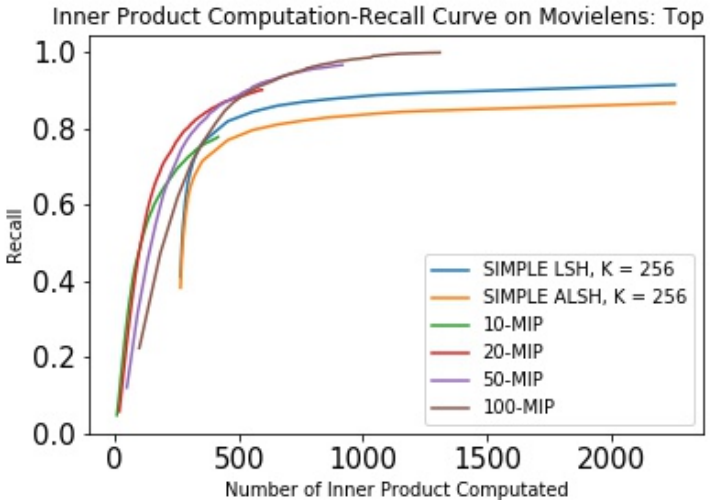


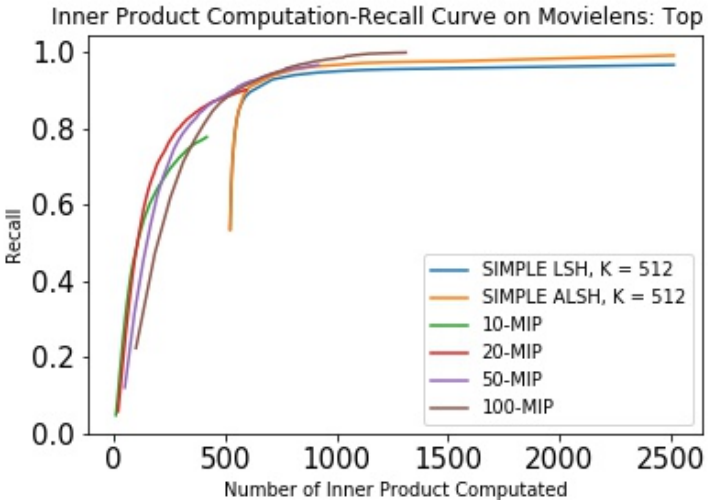
Experiments

K-MIP Graph compared with Simple LSH/ALSH



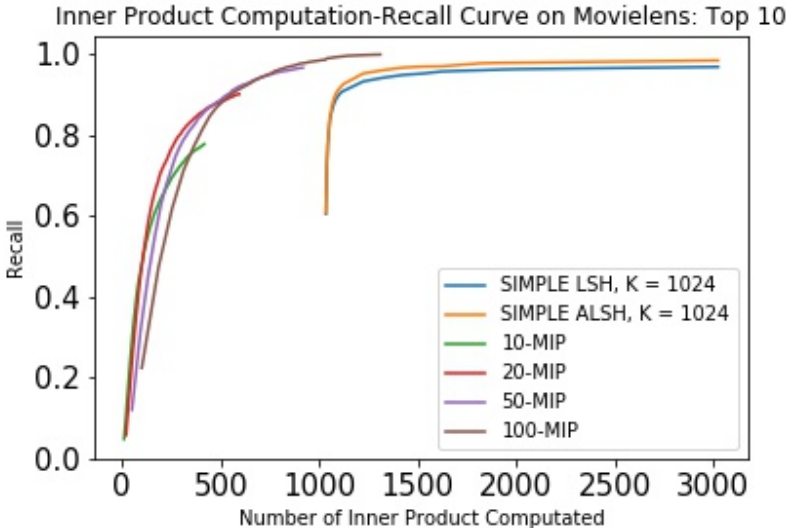


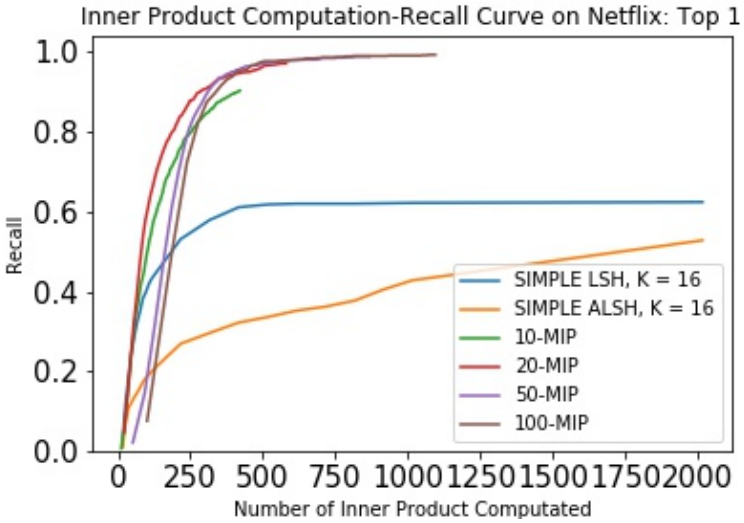


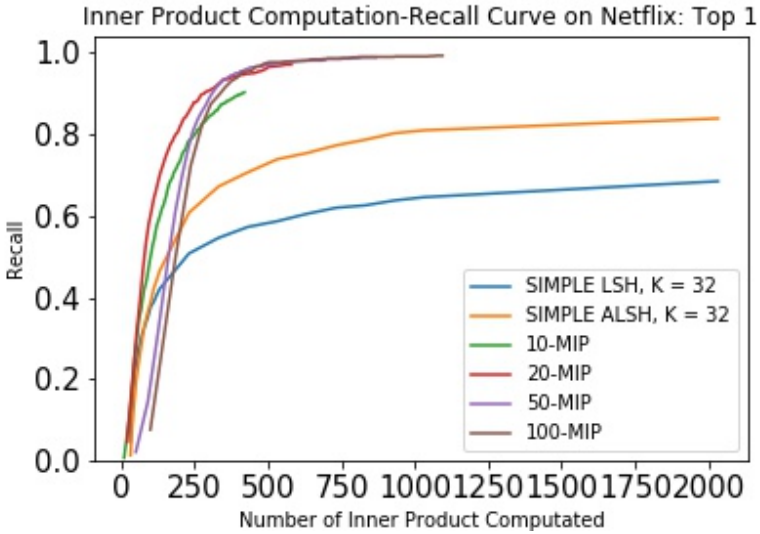


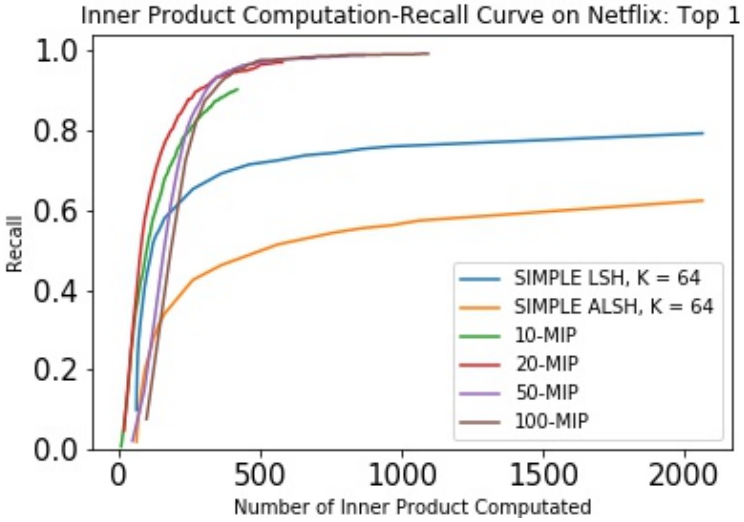
Experiments

K-MIP Graph compared with Simple LSH/ALSH



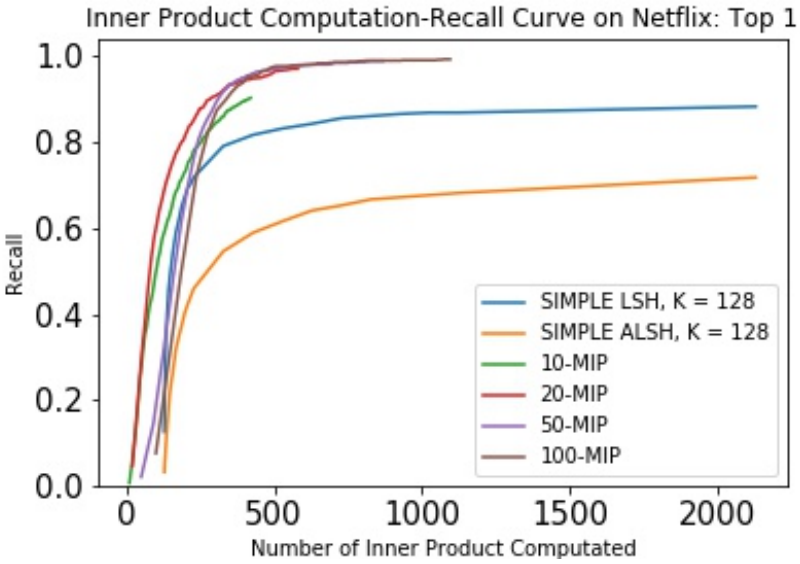


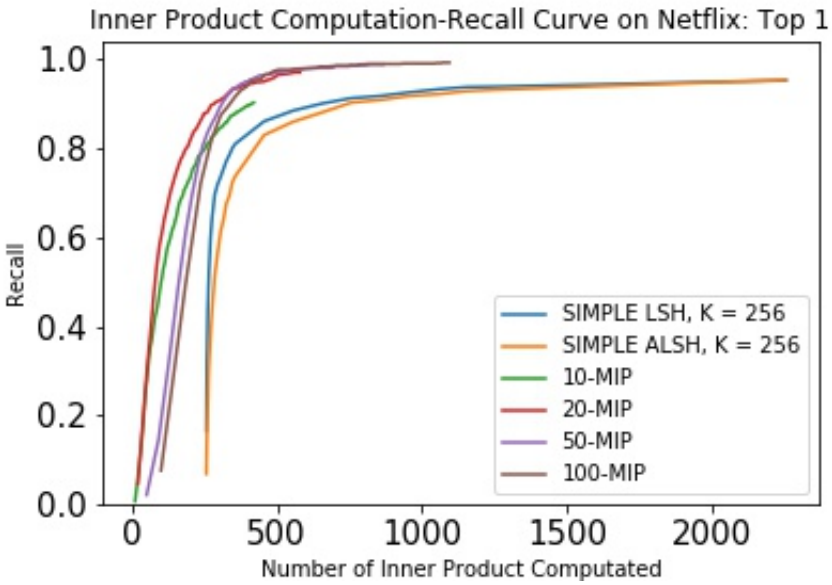


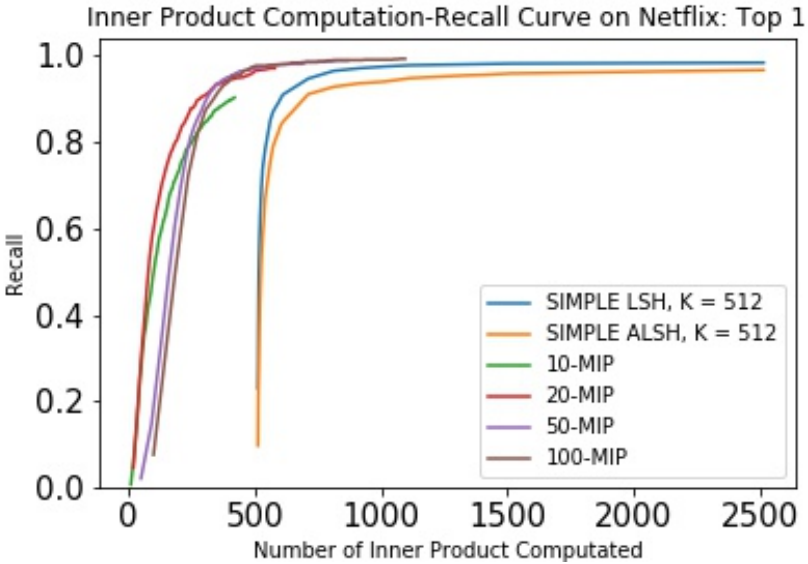


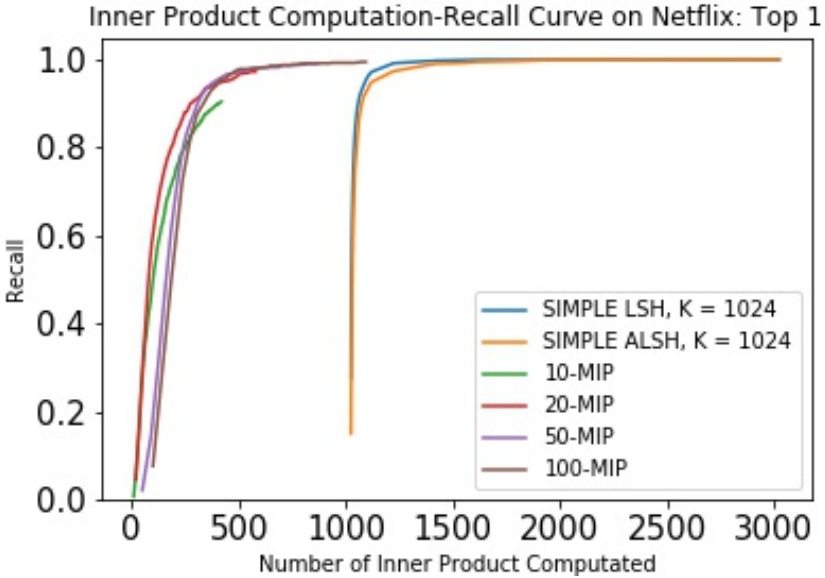
Experiments

K-MIP Graph compared with Simple LSH/ALSH



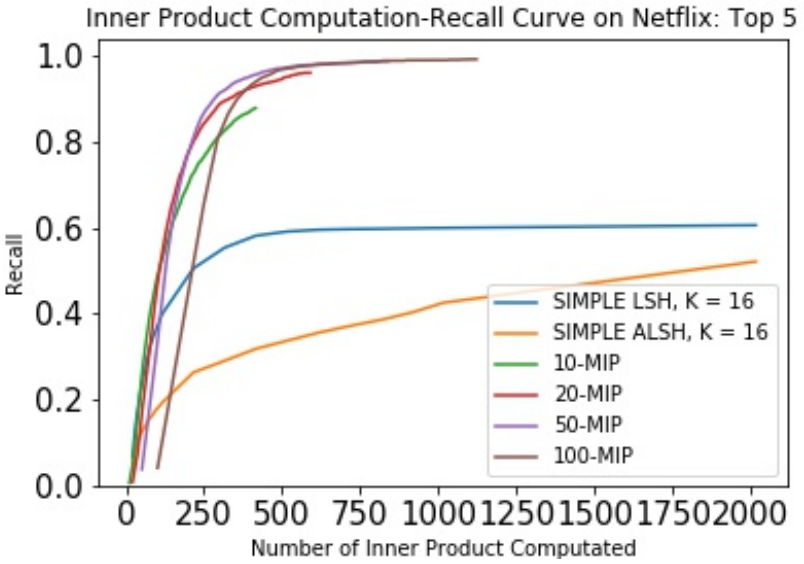


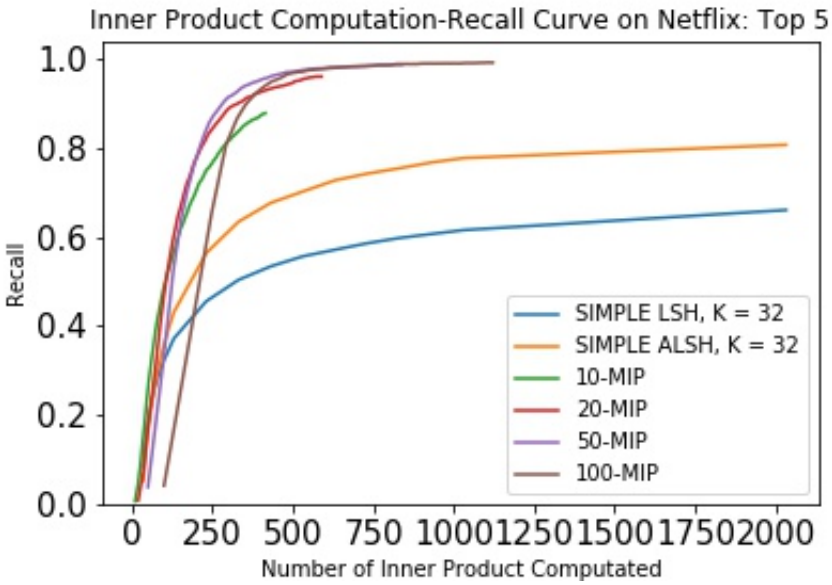


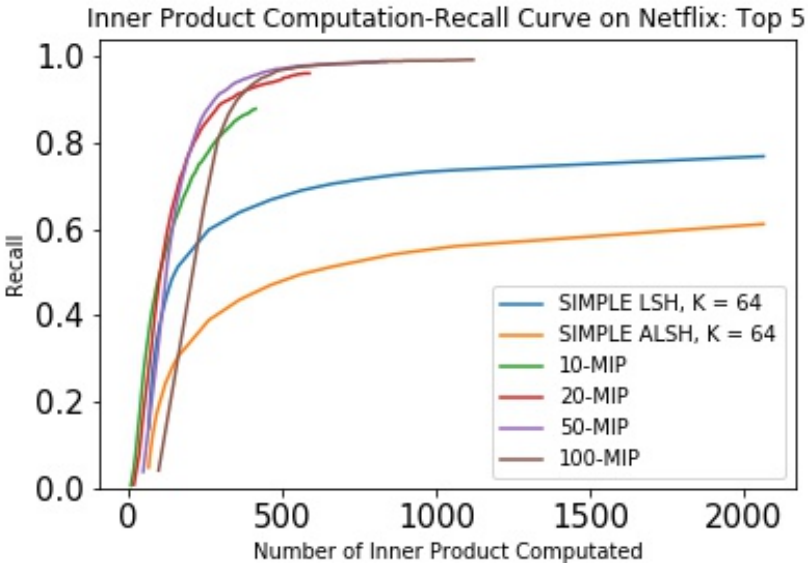


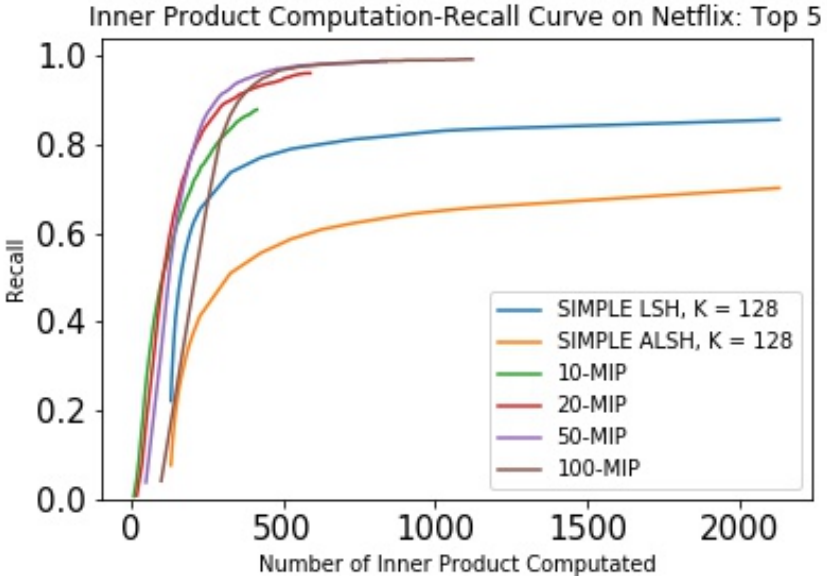
Experiments

K-MIP Graph compared with Simple LSH/ALSH



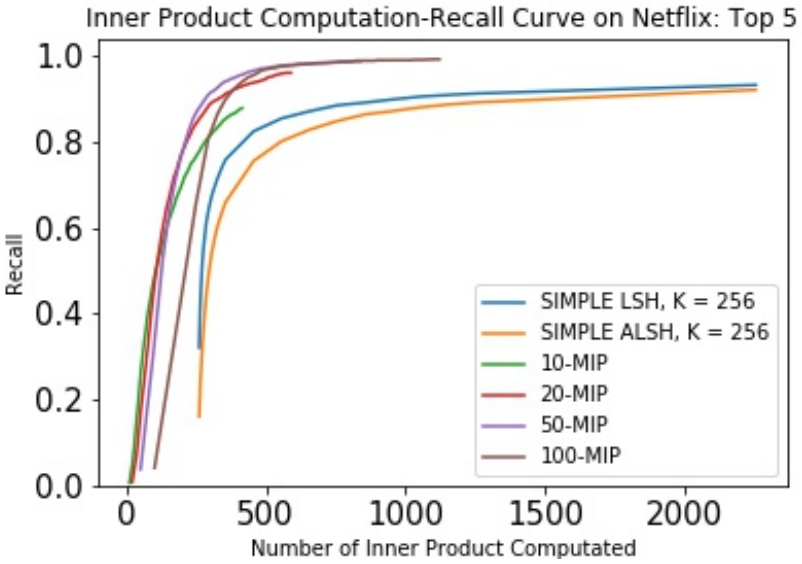


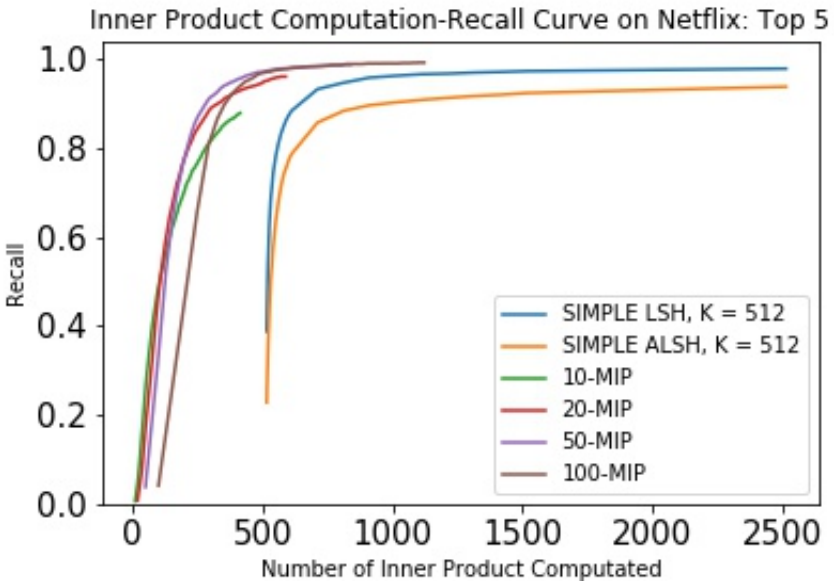


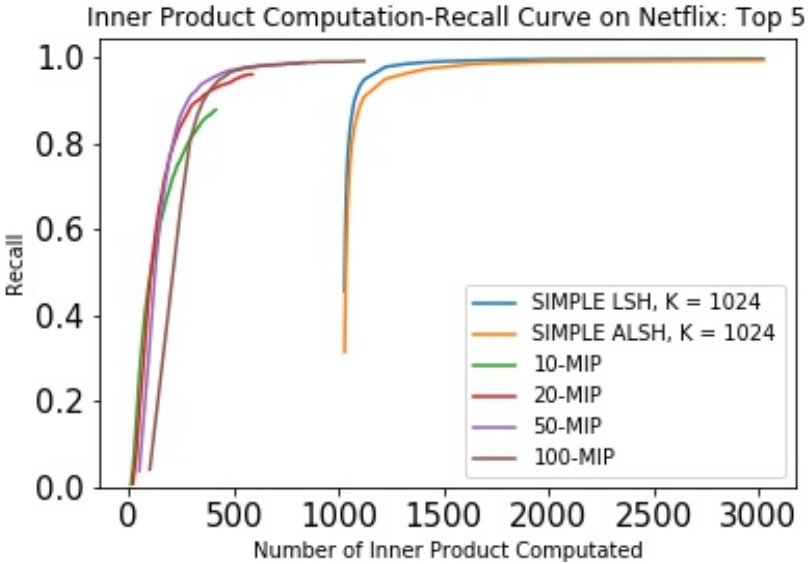


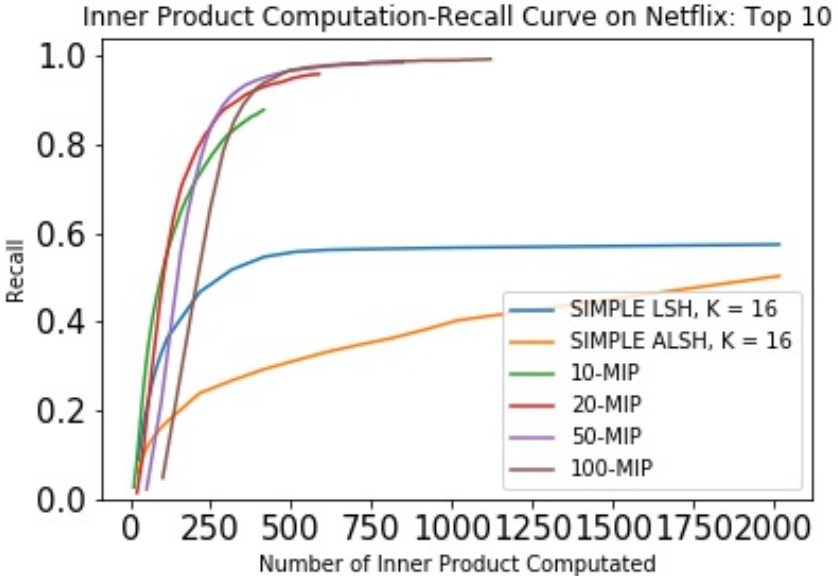
Experiments

K-MIP Graph compared with Simple LSH/ALSH



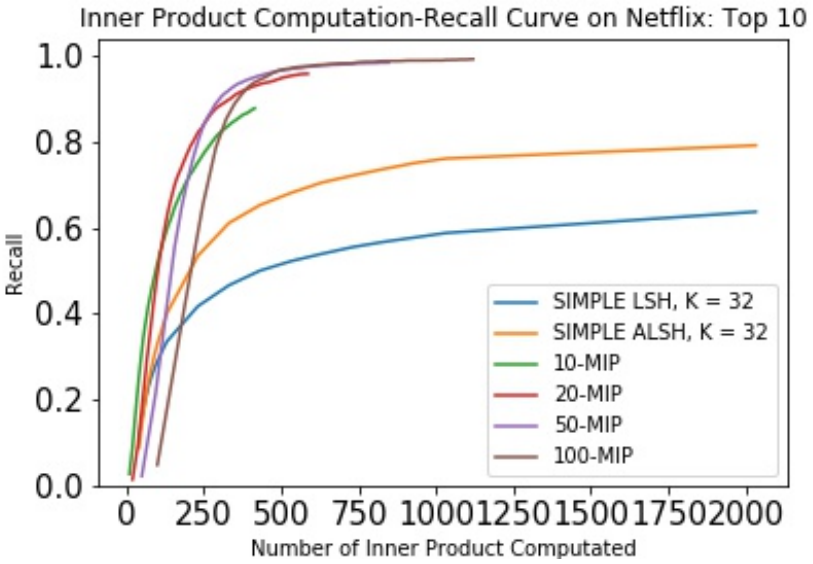


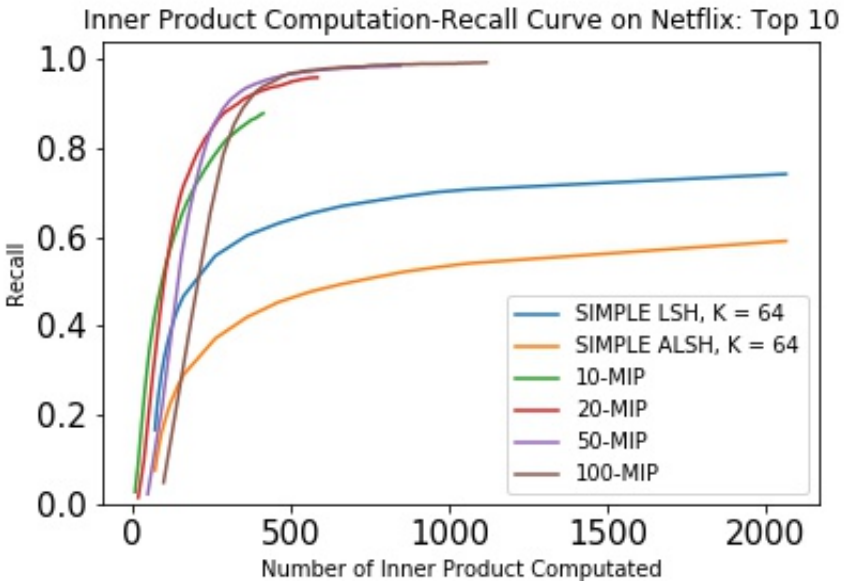


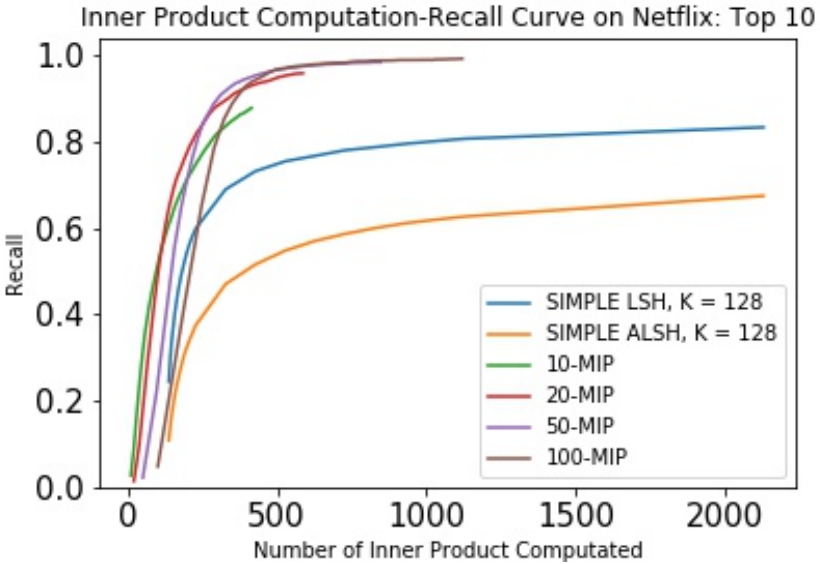


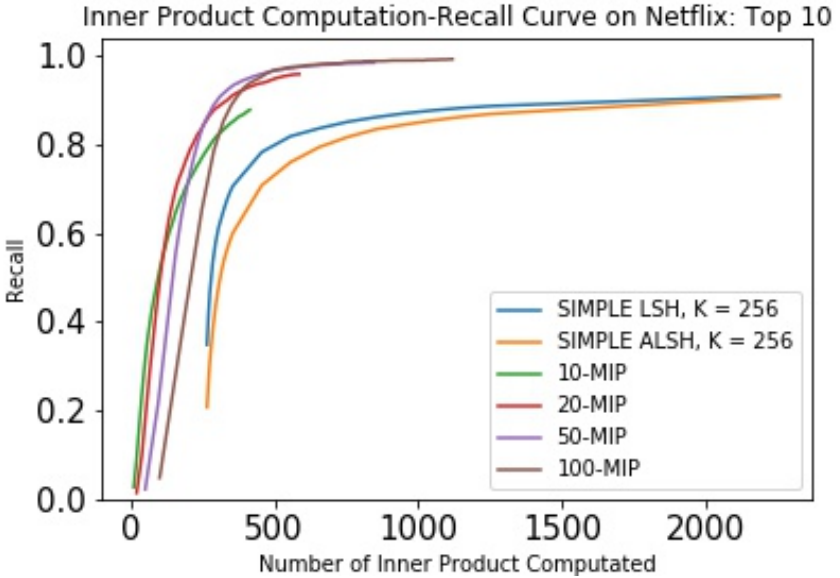
Experiments

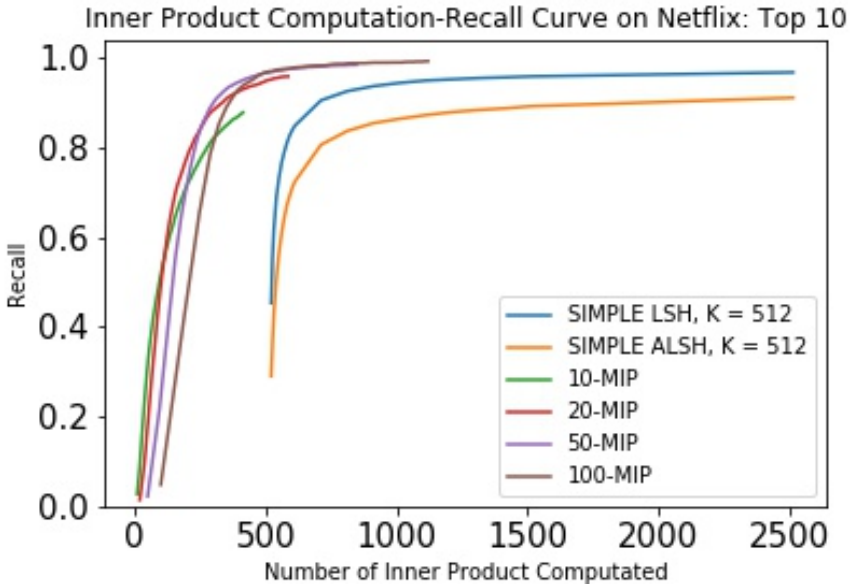
K-MIP Graph compared with Simple LSH/ALSH

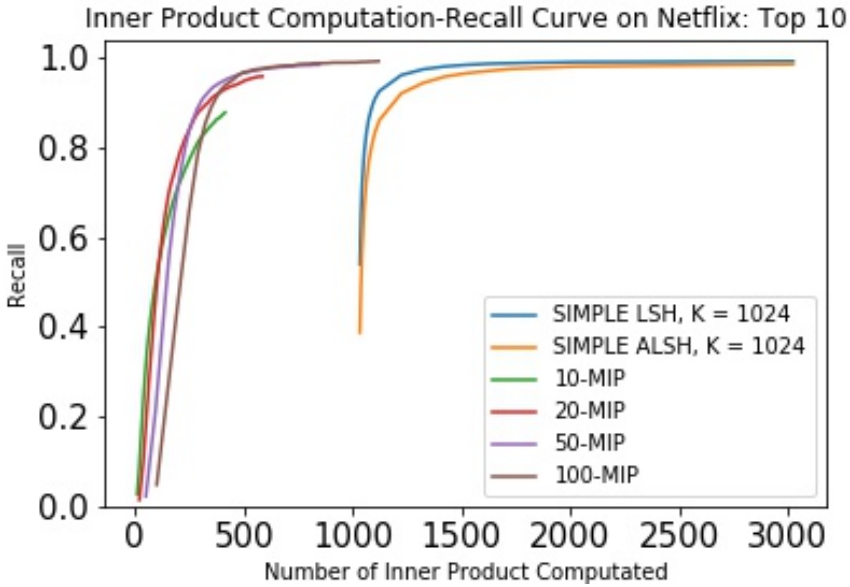









Inner Product Computation-Recall Curve on Netflix: Top 10
Recall
Number of Inner Product Computated
SIMPLE LSH, K = 512
SIMPLE ALSH, K = 512
10-MIP
20-MIP
50-MIP
100-MIP


Inner Product Computation-Recall Curve on Netflix: Top 10
Recall
Number of Inner Product Computated
SIMPLE LSH, K = 1024
SIMPLE ALSH, K = 1024
10-MIP
20-MIP
50-MIP
100-MIP

Discussion

K-MIP Graph compared with Simple LSH/ALSH

Cons of K-MIP Graph:

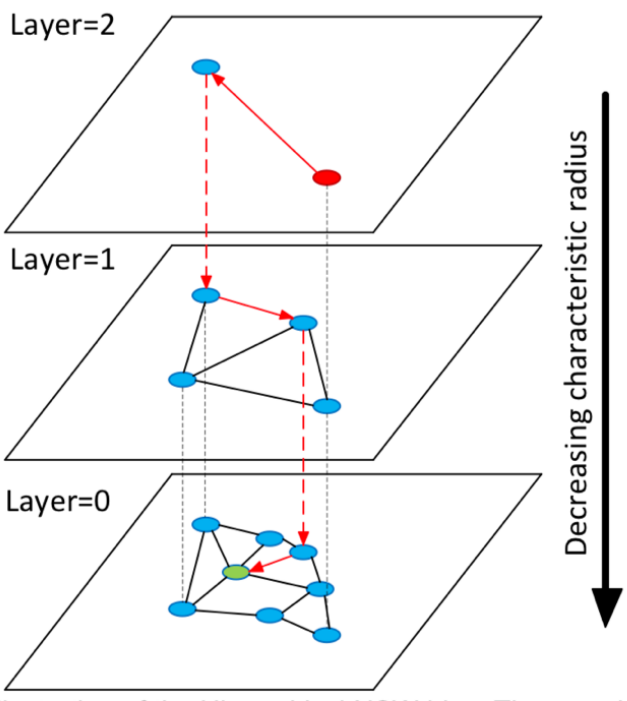
1. For small K, connectivity of K-MIP graph is not guaranteed.
2. Performance is not guaranteed while (LSH is guaranteed to have good performance).
3. For big K, detour can happen during searching and it leads to longer querying time.
4. Construction for the graph is time consuming.

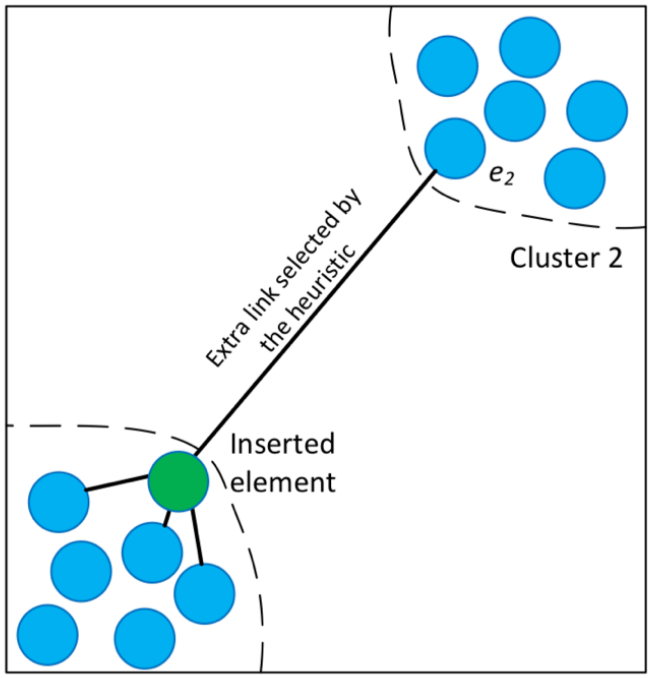
Pros of K-MIP Graph:

1.Faster than LSH on some datasets. (Not repeatable)

Future work

1. To reduce dysconnectivity, we can Separate the links according to their length scale by repeatedly doing KMeans clustering on the dataset. This idea originated from Hierarchical Navigable Small World Graphs (Y. Malkov and D. Yahunin)





Future work

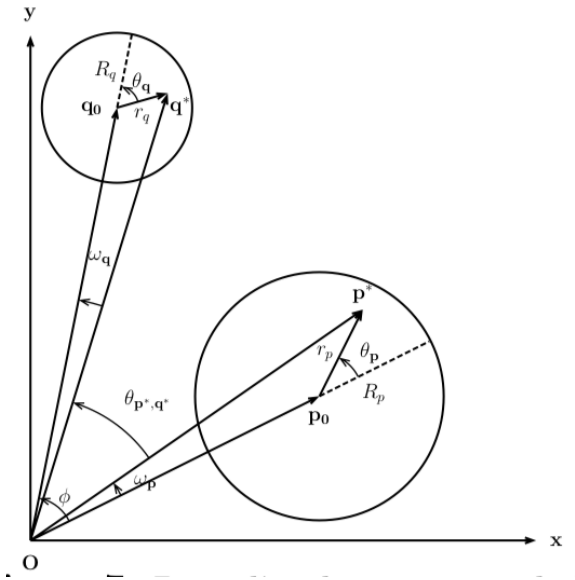
2. Navigating Spreading-out Graph (C. Fu et al)
First build an approxiamte K-MIP graph. Then we incrementally revise the graph with pruning to avoid detour and save storage. At last we run the algorithm for data points not in the graph to ensure connectivity.

3. Build K-MIP graph with data coming in batch. We adopt the idea from Maximum Inner Product using dual cone/ball trees (P. Ram and G. Gray)

Basic Idea: Split dataset using balls/cones. Do exact search based on heuristics.
Main contribution: Fast exact search when queries come in batch. Slower than LSH for a single query

$$
\begin{aligned}
\langle q^*, p^* \rangle &\doteq \langle q_0, p_0 \rangle + r_p r_q \cos(\phi - (\theta_p + \theta_q)) \\
&\quad + r_p \|q_0\| \cos(\phi - \theta_p) + r_q \|p_0\| \cos(\phi - \theta_q) \\
&\leq \max_{\theta_p, \theta_q} \langle q_0, p_0 \rangle + r_p r_q \cos(\phi - (\theta_p + \theta_q)) \\
&\quad + r_p \|q_0\| \cos(\phi - \theta_p) + r_q \|p_0\| \cos(\phi - \theta_q) \\
&\leq \max_{r_p, r_q} \langle q_0, p_0 \rangle + r_p r_q + r_q \|p_0\| + r_p \|q_0\| \\
&\leq \langle q_0, p_0 \rangle + R_p R_q + R_q \|p_0\| + R_p \|q_0\| ,
\end{aligned}
$$

□



Literature Review

LSH to solve (r, ϵ)-Neighbor problem

A family $\mathcal{H}$ of functions from S to U is called (r_1, r_2, p_1, p_2)-sensitive for $D(\bullet, \bullet)$ if for any $p, q \in S$

- if $p \in B(q, r_1)$, then $Pr_{\mathcal{H}}[h(q) = h(p)] \geq p_1$
- if $p \notin B(q, r_2)$, then $Pr_{\mathcal{H}}[h(q) = h(p)] \leq p_2$

(r, ϵ)-Neighbor problem: Determine whether there exists a point $p \in B(q, r)$, or whether all points are from$P - B(q, (1 + \epsilon)r)$.

The LSH algorithm correctly solves the (r, ϵ)-Neighbor problem with the following properties hold.

P1. If there exists p s.t. $p \in B(q, r)$, then $g_j(p) = g_j(q)$ for some $j = 1,2,3..,l$

P2. The total number points to $g_j(p) = g_j(q)$ where p is from $P - B(q, (1 + \epsilon)r)$ is less than bl. (here we can take $b = 3$)

We define $\frac{\log p_1}{\log p_2} = \rho < 1$ (the hashing quality). With $k = -\log_{p_2} n$ and $l = n^{-\rho}$ we guarantee that properties P1 and P2 hold with probability at least $\frac{2}{3} - \frac{1}{e}$.

Pf.

Here we consider $g_i(p) = (h_{i_1}(p), h_{i_2}(p), \ldots, h_{i_k}(p))$, $i = 1,2,...,l$, and h is sampled from $\mathscr{H}$ with replacement.

- If $p \notin B(q, r_2)$, then we set $Pr_{\mathscr{H}}[g(q) = g(p)] \leq p_2^k = \frac{1}{n}$.

Then we have $E[\mathbb{I}(g(q) = g(p))] \leq = 1$ and thus, $E[\sum_i \mathbb{I}(g_i(q) = g_i(p))] \leq = l$. Where $\mathbb{I}(x = y) = 1$ if $x = y$, 0 otherwise.

By Markov inequality: $P[\sum_i \mathbb{I}(g_i(q) = g_i(p)) \geq 3l] \leq \frac{1}{3}$. (other integers can also do)

We denote $\{\sum_i \mathbb{I}(g_i(q) = g_i(p)) < 3l\}$ as event A. $P(B) \geq \frac{2}{3}$ (1)

- If $p \in B(q, r_1)$, then $Pr_{\mathcal{H}}[g_i(q) = g_i(p)] \geq p_1^k = p_1^{\log_{p_2} \frac{1}{n}} = (\frac{1}{n})^{\frac{\log p_1}{\log p_2}} = n^{-\rho}$

If $p \in B(q, r_1)$, then $Pr_{\mathcal{H}}[g(q) \neq g(p)] \leq (1 - n^{-\rho})^l = (1 - n^{-\rho})^{n^\rho} \leq \frac{1}{e}$.

We denote $g_i(q) = g_i(p)$ for some i as event B. $P(B) \geq 1 - \frac{1}{e}$. (2)

So $P(A)P(B) \geq P(A) + P(B) - 1 \geq \frac{2}{3} - \frac{1}{e}$

Note that for (1) to be useful we need l small by making ρ small.

Also note $(1 - n^{-\rho})^{n^\rho}$ is decreasing with ρ so we need small ρ to make the lower bound in (2) big.

Meanwhile with $l = n^\rho$ we also need small ρ to reduce time complexity $O(n^\rho \log n)$ and space complexity $O(n^{1+\rho})$.