SEARCH FOR JOINABLE TABLES IN DATA LAKES

Erkang (Eric) Zhu, PhD Candidate
University of Toronto
ABOUT ME

No, I didn’t start programming when I was 9 (or in high school)

U of Toronto undergrad in Engineering

Then PhD... no regret!
ABOUT ME

Intern at DMX Group @ MSR in 2016
ABOUT ME

Intern at DMX Group @ MSR in 2016

Winner of DMX Cup
Traditional Enterprise Analytics — Heavy IT

Data Scientist

Extract, Transform, Load

ETL Developer

Data

Blah blah
Blah Blah

Database Administrator

Data

Intelligence, Insights
Data Lake Solution — Minimal IT

Data Scientist

Intelligence, Insights

Data

Really?

Azure

amazon web services
Question: More information about these companies?

<table>
<thead>
<tr>
<th>Company</th>
<th>Headquarter</th>
<th>Ticker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alphabet Inc.</td>
<td>Mountain View</td>
<td>GOOGL (NASDAQ)</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>Palo Alto</td>
<td>AAPL (NASDAQ)</td>
</tr>
<tr>
<td>Alibaba</td>
<td>Hangzhou</td>
<td>BABA (NYSE)</td>
</tr>
</tbody>
</table>
Question: More information about these companies?

Query: Tables that can be joined with my own table?

<table>
<thead>
<tr>
<th>Company</th>
<th>Headquarters</th>
<th>Ticker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alphabet Inc.</td>
<td>Mountain View</td>
<td>GOOGL (NASDAQ)</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>Palo Alto</td>
<td>AAPL (NASDAQ)</td>
</tr>
<tr>
<td>Alibaba</td>
<td>Hangzhou</td>
<td>BABA (NYSE)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Task: write ad hoc data transformation script.

<table>
<thead>
<tr>
<th>Company</th>
<th>Headquarter</th>
<th>Ticker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alphabet Inc.</td>
<td>Mountain View</td>
<td>GOOGL (NASDAQ)</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>Palo Alto</td>
<td>AAPL (NASDAQ)</td>
</tr>
<tr>
<td>Alibaba</td>
<td>Hangzhou</td>
<td>BABA (NYSE)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exchange</th>
<th>Ticker</th>
<th>Open</th>
<th>Close</th>
<th>Market Cap</th>
</tr>
</thead>
<tbody>
<tr>
<td>NASDAQ</td>
<td>GOOGL</td>
<td>1,032.50</td>
<td>1,042.32</td>
<td>766 B</td>
</tr>
<tr>
<td>NASDAQ</td>
<td>AAPL</td>
<td>171.10</td>
<td>175.12</td>
<td>892 B</td>
</tr>
<tr>
<td>NYSE</td>
<td>BABA</td>
<td>179.37</td>
<td>174.13</td>
<td>427 B</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Task: write ad hoc data transformation script.

<table>
<thead>
<tr>
<th>Company</th>
<th>Headquarter</th>
<th>Ticker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alphabet Inc.</td>
<td>Mountain View</td>
<td>GOOG (NASDAQ)</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>Palo Alto</td>
<td>AAPL (NASDAQ)</td>
</tr>
<tr>
<td>Alibaba</td>
<td>Hangzhou</td>
<td>BABA (NYSE)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exchange</th>
<th>Ticker</th>
<th>Open</th>
<th>Close</th>
<th>Market Cap</th>
</tr>
</thead>
<tbody>
<tr>
<td>NASDAQ</td>
<td>GOOG</td>
<td>1,032.50</td>
<td>1,042.32</td>
<td>766 B</td>
</tr>
<tr>
<td>NASDAQ</td>
<td>AAPL</td>
<td>171.10</td>
<td>175.12</td>
<td>892 B</td>
</tr>
<tr>
<td>NYSE</td>
<td>BABA</td>
<td>179.37</td>
<td>174.13</td>
<td>427 B</td>
</tr>
</tbody>
</table>

...
“Jack of all trades”

Intelligence, Insights

Data Analytics → Data → Transformation Scripts → Data → Locate Raw Files → Data Lake
CONTRIBUTIONS

Algorithmic solutions that enable data scientists to search for joinable tables and automatically join them — making data scientists more powerful.
JOINABLE TABLE SEARCH

Query Table

<table>
<thead>
<tr>
<th>Company</th>
<th>Headquarters</th>
<th>CEO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Inc.</td>
<td>Mountain View</td>
<td>Sundar Pichai</td>
</tr>
<tr>
<td>Databricks</td>
<td>San Francisco</td>
<td>Ali Ghodsi</td>
</tr>
<tr>
<td>Microsoft Corporation</td>
<td>Redmond</td>
<td>Satya Nadella</td>
</tr>
</tbody>
</table>

Query Column

<table>
<thead>
<tr>
<th>Names</th>
<th>CIK</th>
<th>Q4 Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Inc.</td>
<td>0001288776</td>
<td>26.06 Billions</td>
</tr>
<tr>
<td>Microsoft Corporation</td>
<td>0000789019</td>
<td>23.3 Billions</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Data Lake

Fraction of distinct values in the query that are joinable

Company Set

Names Set
WHY SEARCHING FOR JOINABLE TABLES?

WHY SEARCHING FOR JOINABLE TABLES?

[2] https://open.canada.ca/data/en/dataset/07dcaaf5-6a58-4ce3-8876-e30feef8e8dd
## Why Searching for Joinable Tables?

<table>
<thead>
<tr>
<th>NO\textsubscript{x} Emission</th>
<th>Facility</th>
<th>Postal Code</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>9430.81</td>
<td>Oil Sands</td>
<td>T9H3E3</td>
<td></td>
</tr>
<tr>
<td>50.7</td>
<td>Carberry Factory</td>
<td>R0K0H0</td>
<td></td>
</tr>
<tr>
<td>60.6</td>
<td>Grand Falls</td>
<td>E3Y4A5</td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>Ingleside</td>
<td>K0C1M0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Contributor</th>
<th>Postal Code</th>
<th>Recipient Party</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>E*,B*</td>
<td>T9H3E3</td>
<td>Liberal</td>
<td>397.46</td>
</tr>
<tr>
<td>J*,S*</td>
<td>K0K0H0</td>
<td>Conservative</td>
<td>500.00</td>
</tr>
<tr>
<td>J*,S*</td>
<td>K0K0H0</td>
<td>Conservative</td>
<td>250.00</td>
</tr>
<tr>
<td>M*,S*</td>
<td>K0C1M0</td>
<td>Liberal</td>
<td>400.00</td>
</tr>
</tbody>
</table>


Discover new features for modeling, prediction, or insights
Interactive Navigation of Open Data Lake
A JOINABLE TABLE SEARCH SYSTEM

Tables → Sets → Search Index

Preprocessing (e.g. extract sets) → Indexing
A JOINABLE TABLE SEARCH SYSTEM

1. Tables
2. Preprocessing (e.g. extract sets)
3. Sets
4. Indexing
5. Search Index
6. Column and Table IDs
7. Query
8. Query Set
9. Preprocessing (e.g. extract set)
10. Table
11. User Interface
A JOINABLE TABLE SEARCH SYSTEM

Tables → Sets → Search Index → User Interface

Preprocessing (e.g. extract sets) → Indexing → Column and Table IDs

Query → Query Set → Preprocessing (e.g. extract set)

Navigating Open Data Linkages, Best Demo (VLDB 17')
A JOINABLE TABLE SEARCH SYSTEM

- Tables
  - Preprocessing (e.g. extract sets)
  - Sets
  - Indexing
  - Search Index
  - Column and Table IDs
  - Query
  - Query Set
  - Table

Set Search Indexes
- LSH Ensemble (VLDB 16')

Navigating Open Data Linkages, Best Demo (VLDB 17')

User Interface
A JOINABLE TABLE SEARCH SYSTEM

Sets \rightarrow \text{Search Index} \rightarrow \text{Query Set}

- Set Search Indexes
  - LSH Ensemble (VLDB 16')

Indexing

Column and Table IDs

Query
DATA SKETCHES

Challenges:
- A set from a table column can have tens of thousands of values
- A data lake can have hundreds of thousands of tables
- An Open Data lake: 200K tables, and Max set size: 22M

Data sketches for sets:
- Small, fixed-size summaries (a few KBs)
- Probabilistic guarantee (more on this later)
- $O(|X|)$ time to build, $|X|$ is set size

We use MinHash\(^1\) data sketches for joinable table search

---

MINHASH

Random, independent hash functions (e.g., MurmurHash3)

\[
\begin{align*}
  h_1 & : \{ "ATL", "LAX", "ORD", "DFW", "DEN", "JFK", "SFO", "LAS", "SEA", "CLT" \}
\end{align*}
\]
MINHASH

Random, independent hash functions (e.g., MurmurHash3)


0x123, 0x3f1, 0x1c, 0x312, 0x456, 0x42a, 0xb12, 0x98a, 0x547, 0x132

<table>
<thead>
<tr>
<th>Hash Function</th>
<th>Min Hash Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_1$</td>
<td>0x123</td>
</tr>
<tr>
<td>$h_2$</td>
<td></td>
</tr>
<tr>
<td>$h_3$</td>
<td></td>
</tr>
<tr>
<td>$h_4$</td>
<td></td>
</tr>
<tr>
<td>$h_5$</td>
<td></td>
</tr>
<tr>
<td>$h_6$</td>
<td></td>
</tr>
<tr>
<td>$h_7$</td>
<td></td>
</tr>
<tr>
<td>$h_8$</td>
<td></td>
</tr>
</tbody>
</table>
MINHASH

Random, independent hash functions (e.g., MurmurHash3)

\{"ATL", "LAX", "ORD", "DFW", "DEN", "JFK", "SFO", "LAS", "SEA", "CHF2"\}

<table>
<thead>
<tr>
<th>Hash Function</th>
<th>Min Hash Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_1$</td>
<td>0x123</td>
</tr>
<tr>
<td>$h_2$</td>
<td>0x14α</td>
</tr>
<tr>
<td>$h_3$</td>
<td></td>
</tr>
<tr>
<td>$h_4$</td>
<td></td>
</tr>
<tr>
<td>$h_5$</td>
<td></td>
</tr>
<tr>
<td>$h_6$</td>
<td></td>
</tr>
<tr>
<td>$h_7$</td>
<td></td>
</tr>
<tr>
<td>$h_8$</td>
<td></td>
</tr>
</tbody>
</table>

$0x25b$, $0xa21$, $0x81a$, $0x514$, $0xb51$, $0x431$, $0x14a$, $0x981$, $0x1c7$, $0x16b$
MINHASH

Random, independent hash functions (e.g., MurmurHash3)

\{'ATL', 'LAX', 'ORD', 'DFW', 'DEN', 'JFK', 'SFO', 'LAS', 'SEA', 'CLT'\}

<table>
<thead>
<tr>
<th>Hash Function</th>
<th>Min Hash Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_1$</td>
<td>0x123</td>
</tr>
<tr>
<td>$h_2$</td>
<td>0x14a</td>
</tr>
<tr>
<td>$h_3$</td>
<td>0x41c</td>
</tr>
<tr>
<td>$h_4$</td>
<td>0x1bc</td>
</tr>
<tr>
<td>$h_5$</td>
<td>0x2c3</td>
</tr>
<tr>
<td>$h_6$</td>
<td>0x5d1</td>
</tr>
<tr>
<td>$h_7$</td>
<td>0x613</td>
</tr>
<tr>
<td>$h_8$</td>
<td>0x312</td>
</tr>
</tbody>
</table>

[Optional]: speed up using only one hash function and universal hashing
MINHASH

For two sets $X$ and $Y$, the probability of a minimum hash value collision:

$$P[\min(h_1(X)) = \min(h_1(Y))] = \frac{|X \cap Y|}{|X \cup Y|}$$

This is Jaccard similarity

<table>
<thead>
<tr>
<th>$X$</th>
<th>Hash Function</th>
<th>Min Hash Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_1$</td>
<td>0x123</td>
<td></td>
</tr>
<tr>
<td>$h_2$</td>
<td>0x14a</td>
<td></td>
</tr>
<tr>
<td>$h_3$</td>
<td>0x41c</td>
<td></td>
</tr>
<tr>
<td>$h_4$</td>
<td>0x1bc</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$Y$</th>
<th>Hash Function</th>
<th>Min Hash Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_1$</td>
<td>0x123</td>
<td></td>
</tr>
<tr>
<td>$h_2$</td>
<td>0x14a</td>
<td></td>
</tr>
<tr>
<td>$h_3$</td>
<td>0x256</td>
<td></td>
</tr>
<tr>
<td>$h_4$</td>
<td>0x78d</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
MINHASH

We can use hash tables to find “approximately” joinable columns under Jaccard similarity.

<table>
<thead>
<tr>
<th>Hash Function</th>
<th>Min Hash Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_1$</td>
<td>0x123</td>
</tr>
<tr>
<td>$h_2$</td>
<td>0x14a</td>
</tr>
<tr>
<td>$h_3$</td>
<td>0x41c</td>
</tr>
<tr>
<td>$h_4$</td>
<td>0x1bc</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hash Function</th>
<th>Min Hash Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_1$</td>
<td>0x123</td>
</tr>
<tr>
<td>$h_2$</td>
<td>0x14a</td>
</tr>
<tr>
<td>$h_3$</td>
<td>0x256</td>
</tr>
<tr>
<td>$h_4$</td>
<td>0x78d</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Column $X$  

Hash Tables

Column $Q$
MINHASH LSH

Locality Sensitive Hashing (LSH) index provides better accuracy through "boosting"

<table>
<thead>
<tr>
<th>Hash Function</th>
<th>Min Hash Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_1$</td>
<td>0x123</td>
</tr>
<tr>
<td>$h_2$</td>
<td>0x14a</td>
</tr>
<tr>
<td>$h_3$</td>
<td>0x41c</td>
</tr>
<tr>
<td>$h_4$</td>
<td>0x1bc</td>
</tr>
<tr>
<td>$h_5$</td>
<td>0x2c3</td>
</tr>
<tr>
<td>$h_6$</td>
<td>0x5d1</td>
</tr>
<tr>
<td>$h_7$</td>
<td>0x613</td>
</tr>
<tr>
<td>$h_8$</td>
<td>0x312</td>
</tr>
</tbody>
</table>

Hash Keys $r = 4$

[0x123, 0x14a, 0x41c, 0x1bc]

Hash Tables $b = 2$

$H_1$

$H_2$

MINHASH LSH

\[ P[\exists i, H_i(X) = H_i(Q)] \]

- Probability of retrieving from LSH

\[ \frac{|X \cap Q|}{|X \cup Q|} \]

- Jaccard similarity

Threshold

False negative probability

False positive probability
MINHASH LSH

\[ P[\exists i, H_i(X) = H_i(Q)] \]

Probability of retrieving from LSH

\[ \frac{|X \cap Q|}{|X \cup Q|} \]

Jaccard similarity

Threshold

Without using LSH

False negative probability

False positive probability
MINHASH LSH

Minimize error probability with respect to the Jaccard similarity threshold, by tuning.

\[ P[\exists i, H_i(X) = H_i(Q)] \]

Probability of retrieving from LSH

\[ \frac{|X \cap Q|}{|X \cup Q|} \]

Jaccard similarity

False negative probability

False positive probability
IS JACCARD SIMILARITY THE RIGHT MEASURE?

As $|X_1 \cap Q| = |X_2 \cap Q|$, both $X_1$ and $X_2$ are equally good for joining with $Q$. 

[Diagram showing Venn diagrams of $Q$, $X_1$, and $X_2$.]
IS JACCARD SIMILARITY THE RIGHT MEASURE?

As $|X_1 \cap Q| = |X_2 \cap Q|$, both $X_1$ and $X_2$ are equally good for joining with $Q$.

But $\frac{|X_1 \cap Q|}{|X_1 \cup Q|} > \frac{|X_2 \cap Q|}{|X_2 \cup Q|}$

Alternatively, $\frac{|X_1 \cap Q|}{|Q|} = \frac{|X_2 \cap Q|}{|Q|}$
CONTAINMENT SEARCH

We use containment, $\frac{|x \cap q|}{|q|}$, as a measure for joinable tables.

We can “fix” MinHash LSH for containment.
CONTAINMENT SEARCH

We use containment, $\frac{|X \cap Q|}{|Q|}$, as a measure for joinable tables.

We can “fix” MinHash LSH for containment.

LSH Ensemble$^\dagger$

- Part of datasketch Python library$^2$, together with MinHash LSH
- The library is used by Google (TimeSketch), MIT (Aurum Data Discovery) and Stanford (NLP)
- Over 760 stars on Github

---

$^\dagger$ Erkang Zhu, Fatemeh Nargesian, Ken Pu, Renée J. Miller, “LSH Ensemble: L Internet-Scale Domain Search”, VLDB 2016

$^2$ https://github.com/ekzhu/datasketch
FROM CONTAINMENT TO JACCARD

Apply Inclusion-Exclusion Principle:

\[
\frac{|X \cap Q|}{|X \cup Q|} = \frac{|X \cap Q|}{|X| + |Q| - |X \cap Q|} = 1 + \frac{|X \cap Q|}{|Q|} - \frac{|X \cap Q|}{|Q|}
\]
FROM CONTAINMENT TO JACCARD

A containment threshold \((t^*)\) can be translated into a Jaccard threshold \((s^*)\)

\[
s^*_{|X|} = \frac{t^*}{1 + \frac{|X|}{|Q|} - t^*}
\]
FROM CONTAINMENT TO JACCARD

A containment threshold \((t^*)\) can be translated into a Jaccard threshold \((s^*)\)

\[
S_{|X|}^* = \frac{t^*}{1 + \frac{|X|}{|Q|} - t^*}
\]

Internal Jaccard threshold \(s_{|X|}^*\)

Tune MinHash LSH index using \(S_{|X|}^*\)
FROM CONTAINMENT TO JACCARD

A containment threshold \((t^*)\) can be translated into a Jaccard threshold \((s^*)\)

\[
S^*_{|X|} = \frac{t^*}{1 + \frac{|X|}{|Q|} - t^*}
\]

Internal Jaccard threshold \(S^*_{|X|}\)

\(|X|\) is not a constant!

Alternative: use \(u\) (the largest set size) to approximate \(|X|\)

\[
S^*_u = \frac{t^*}{1 + \frac{u}{|Q|} - t^*}
\]
False positives caused by approximation

Correct threshold:
\[ S_{|X|}^* = \frac{t^*}{1 + \frac{|X|}{|Q|} - t^*} \]

Approximate threshold:
\[ S_u^* = \frac{t^*}{1 + \frac{u}{|Q|} - t^*} \]
Correct threshold

$S^*_{|x|} = \frac{t^*}{1 + \frac{|x|}{|Q|} - t^*}$

False positives caused by approximation

Approximate threshold

$S^*_u = \frac{t^*}{1 + \frac{u}{|Q|} - t^*}$
REDUCE FALSE POSITIVES — BRUTE FORCE

We can verify the exact containment of all sets in the output — “posting processing”
REDUCE FALSE POSITIVES — BRUTE FORCE

We can verify the exact containment of all sets in the output — “posting processing”

https://www.theguardian.com/lifeandstyle/2015/dec/02/eat-drink-bad-for-us-evolution
Set sizes in the index:

\{3, 3, 3, 3, 4, 4, 4, 99, 99, 99, 100, 100, 100\}
Set sizes in the index:
\{3, 3, 3, 4, 4, 4, 99, 99, 99, 100, 100, 100\}

\( u = 100, \ |X| = 3 \)

Correct threshold: \( s_{|X|}^* \)

Approximate threshold: \( s_{u}^* \)
Set sizes in the index:

\{3, 3, 3, 4, 4, 4, 99, 99, 99, 100, 100, 100\}

\[ u = 100, \ |X| = 3 \]

Correct threshold: \( S_{|X|^*} \)

Approximate threshold: \( s_{u^*} \)

Too many false positives to remove by post-processing
Use two indexes:

**Set sizes in index 1**

\{3, 3, 3, 4, 4, 4, 4\}

**Set sizes in index 2**

\{99, 99, 99, 100, 100, 100\}
Use two indexes:

Set sizes in index 1
\{3, 3, 3, 4, 4, 4, 4\}

Set sizes in index 2
\{99, 99, 99, 100, 100, 100\}

\(u_1 = 4, |X| = 3\)
Use two indexes:

Set sizes in index 1:
\{3, 3, 3, 3, 4, 4, 4, 4\}

\(u_1 = 4, \ |X| = 3\)

Correct threshold: \(s^{*}_{|X|}\)

Approximate threshold: \(s^{*}_u\)

Set sizes in index 2:
\{99, 99, 99, 100, 100, 100\}

\(u_2 = 100, \ |X| = 99\)

Correct threshold: \(s^{*}_{|X|}\)

Approximate threshold: \(s^{*}_u\)
Correct threshold
\( S^*_|X| = \frac{t^*}{1 + |X|/|Q| - t^*} \)

Approximate threshold
\( S^*_u = \frac{t^*}{1 + u/|Q| - t^*} \)

False positives caused by approximation

Jaccard Threshold \( s^* \)

Containment Threshold \( t^* \)
REDUCE FALSE POSITIVES — BRUTE FORCE

We can verify the exact containment of all sets in the output — “posting processing”
Use two indexes:

Set sizes in index 1: \( \{3, 3, 3, 3, 4, 4, 4, 4\} \)

Set sizes in index 2: \( \{99, 99, 99, 100, 100, 100\} \)

\( u_1 = 4, |X| = 3 \)

\( u_2 = 100, |X| = 99 \)
REDUCE FALSE POSITIVES - PARTITIONS

Partition the sets into \( n \) disjoint partitions with increasing set sizes:

\[
\begin{align*}
\text{MinHash LSH} \\
[\ell, u]
\end{align*}
\]
REDUCE FALSE POSITIVES - PARTITIONS

Partition the sets into $n$ disjoint partitions with increasing set sizes:
REDUCE FALSE POSITIVES - PARTITIONS

Partition the sets into $n$ disjoint partitions with increasing set sizes:

\[
\begin{align*}
[l_1, u_1] & \quad \text{MinHash LSH} \\
[l_2, u_2] & \quad \text{MinHash LSH} \\
[l_3, u_3] & \quad \text{MinHash LSH} \\
& \quad \vdots \\
[l_n, u_n] & \quad \text{MinHash LSH}
\end{align*}
\]

Reduce false positive at the rate of $\frac{1}{(u - u_i)^2}$ by using $u_i$ instead of $u$ for each partition!
REDUCE FALSE POSITIVES - PARTITIONS

Partition the sets into \( n \) disjoint partitions with increasing set sizes:

\[
\begin{align*}
&[l_1, u_1] \\
&[l_2, u_2] \\
&[l_3, u_3] \\
&\vdots \\
&[l_n, u_n]
\end{align*}
\]

Reduce false positive at the rate of \( \frac{1}{(u-u_i)^2} \) by using \( u_i \) instead of \( u \) for each partition!

Infinite partitions? Query time grows linearly with \( n \)!
UPPER BOUND NUM. FALSE POSITIVES

Upper bound of number of false positives over a set size interval \([l, u]\)

\[
N_{l,u}^{FP} \leq \sum_{x \in [l, u]} N_x \cdot \left(1 - \frac{x}{u}\right)
\]

Number of sets with size \(x\)

\([l, u]\)
OPTIMAL PARTITIONING — UNIFORM DIST.

Example: \( u = 100, l = 1, N_{l,u} = 100, n = 2 \), assume uniform distribution

\[ l_1 = 1 \quad u_1 \quad u_2 = 100 \]
OPTIMAL PARTITIONING — UNIFORM DIST.

Example: \( u = 100, l = 1, N_{l,u} = 100, n = 2, \) assume uniform distribution

\[ l_1 = 1 \quad \uparrow \quad u_1 \quad u_2 = 100 \]
**OPTIMAL PARTITIONING — UNIFORM DIST.**

Example: \( u = 100, l = 1, N_{l,u} = 100, n = 2 \), assume uniform distribution

\[ l_1 = 1 \quad u_1 \quad u_2 = 100 \]

Number of false positives (upper bound):
\[ N_{l_1,u_1}^{FP} + N_{l_2,u_2}^{FP} \]

Optimal 2-partitioning boundary

 Partition boundary \((u_1)\)
REAL SIZE DISTRIBUTIONS ARE NOT UNIFORM

Distribution of Set Sizes in Canadian Open Data tables
OPTIMAL PARTITIONING FOR ANY DISTRIBUTION

Setting $u_1, u_2, \ldots, u_{n-1}$ to minimize the sum of upper bounds of $n$ partitions:

$$\sum_{i=1}^{n} \left( \sum_{x \in [u_{i-1}, u_i]} N_x \cdot \left(1 - \frac{x}{u_i}\right) \right)$$
OPTIMAL PARTITIONING FOR ANY DISTRIBUTION

Setting \( u_1, u_2, \ldots, u_{n-1} \) to minimize the sum of upper bounds of \( n \) partitions:

\[
\sum_{i=1}^{n} \left( \sum_{x \in [u_i, u_{i+1}]} N_x \cdot \left( 1 - \frac{x}{u_i} \right) \right)
\]

Dynamic programming to the rescue!

Complexity is \( O(|N|^2 \cdot n) \)

- \( N \) is the domain of all set sizes, \( n \) is the number of partitions
Setting \( u_1, u_2, \ldots, u_{n-1} \) to minimize the sum of upper bounds of \( n \) partitions:

\[
\sum_{i=1}^{n} \left( \sum_{x \in [l_i, u_i]} N_x \cdot \left(1 - \frac{x}{u_i}\right) \right)
\]

Dynamic programming to the rescue!

Complexity is \( O(|N|^2 \cdot n) \)
- \( N \) is the domain of all set sizes, \( n \) is the number of partitions

The complexity is quite manageable in practice as: \(|N| \ll |u_n - l_1|\)
- Number of bank accounts \(<|Amount in Jeff Bezos' - Amount in Eric's|
- All of Canada, U.S., and U.K. Open Data: \(|N| = 14K, <10 \text{ min in Python on a laptop}\)
- WDC Web Table: \(|N| = 4K\)
**LSH ENSEMBLE: KEY POINTS**

Given a query set $Q$ and containment threshold $t^*$

- **Indexing**: DP, MinHash LSH on partitions, $s_u^* = \frac{t^*}{\frac{u}{1+|Q|}t^*}$
- **Querying**: probing hash tables; optionally remove false positives
- Also handles arbitrary $|Q|$ and $t^*$
LSH ENSEMBLE: KEY POINTS

Given a query set $Q$ and containment threshold $t^*$

- Indexing: DP, MinHash LSH on partitions, $s_u^* = \frac{t^*}{u} \frac{1}{1 + |Q|} \frac{1}{t^*}$
- Querying: probing hash tables; optionally remove false positives
- Also handles arbitrary $|Q|$ and $t^*$

Parallelize query over partitions

Small query memory footprints
- MinHash is only 1KB (128 hash functions)

Scale to massive number of sets
- Tested using 220 Million sets from WDC Web Table (parallel over 5 machines)
A JOINABLE TABLE SEARCH SYSTEM

- LSH Ensemble (VLDB 16')
Josie vs. LSH Ensemble

They are different!
### JOSIE VS. LSH ENSEMBLE

They are different!

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Accuracy</th>
<th>Search Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSH Ensemble</td>
<td>Approximate</td>
<td>Threshold</td>
</tr>
<tr>
<td>JOSIE</td>
<td>Exact</td>
<td>Top-K</td>
</tr>
</tbody>
</table>
WHEN TO USE TOP-K SEARCH?

Threshold search: user must specify a containment threshold
- User may not know a good threshold
- User may not understand containment

Top-K problem: just return the best $k$ results
- No knowledge of relevance measure is required
- We showed that for small $k$ ($< 20$), our exact top-$k$ algorithm can be faster than LSH Ensemble with decreasing threshold hack!
WHEN TO USE TOP-K SEARCH?

Threshold search: user must specify a containment threshold
  - User may not know a good threshold
  - User may not understand containment

Top-K problem: just return the best $k$ results
  - No knowledge of relevance measure is required
  - We showed that for small $k$ ($< 20$), our exact top-k algorithm can be faster than LSH Ensemble with decreasing threshold hack!
WHEN TO USE TOP-K SEARCH?

Threshold search: user must specify a containment threshold
  ▪ User may not know a good threshold
  ▪ User may not understand containment

Top-K problem: just return the best k results
  ▪ No knowledge of relevance measure is required
  ▪ We showed that for small k (< 20), our exact top-k algorithm can be faster than LSH Ensemble with decreasing threshold hack!

Use top-k for less sophisticated users and small k
TOP-K OVERLAP SET SIMILARITY SEARCH

Overlap as the relevance measure: $|Q \cap X|$, equivalent to containment
INVERTED INDEX — A MATRIX PERSPECTIVE

Domain of data values

\[ Q \]

\[ v_1 \quad v_2 \quad v_3 \quad v_4 \quad v_5 \quad v_6 \quad v_7 \quad v_8 \]

Sets

\[ X_1 \]
\[ X_2 \]
\[ X_3 \]
\[ X_4 \]

A posting list

A set
BASELINES FOR FIND TOP-1

\[ \begin{array}{cccccccc}
   v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 \\
   Q & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\
   X_1 & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\
   X_2 & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\
   X_3 & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\
   X_4 & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\
\end{array} \]

Posting list union\(^1\): reading all posting lists costs 7 values

BASELINES FOR FIND TOP-1

Postion list union$^1$: reading all posting lists costs 7 values

Prefix filtering$^2$: reading candidate sets


**BASELINES FOR FIND TOP-1**

<table>
<thead>
<tr>
<th></th>
<th>$v_1$</th>
<th>$v_2$</th>
<th>$v_3$</th>
<th>$v_4$</th>
<th>$v_5$</th>
<th>$v_6$</th>
<th>$v_7$</th>
<th>$v_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$X_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$X_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$X_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$X_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Posting list union**: reading all posting lists costs 7 values.

**Prefix filtering**: reading candidate sets.


BASELINES FOR FIND TOP-1

$Q \quad v_1 \ v_2 \ v_3 \ v_4 \ v_5 \ v_6 \ v_7 \ v_8$

$X_1$

$X_2$

$X_3$

$X_4$

$Q \quad v_1 \ v_2 \ v_3 \ v_4 \ v_5 \ v_6 \ v_7 \ v_8$

$X_1$

$X_2$

$X_3$

$X_4$

Postponed list union$^1$: reading all posting lists costs 7 values

Prefix filtering$^2$: reading candidate sets

BASELINES FOR FIND TOP-1

Posting list union¹: reading all posting lists costs 7 values

Prefix filtering²: reading candidate sets

---

¹ Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to Information Retrieval. Cambridge University Press, 2008
## Baselines for Find Top-1

<table>
<thead>
<tr>
<th>$Q$</th>
<th>$v_1$</th>
<th>$v_2$</th>
<th>$v_3$</th>
<th>$v_4$</th>
<th>$v_5$</th>
<th>$v_6$</th>
<th>$v_7$</th>
<th>$v_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$X_2$</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$X_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$X_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Posting list union\(^1\): reading all posting lists costs 7 values

<table>
<thead>
<tr>
<th>$Q$</th>
<th>$v_1$</th>
<th>$v_2$</th>
<th>$v_3$</th>
<th>$v_4$</th>
<th>$v_5$</th>
<th>$v_6$</th>
<th>$v_7$</th>
<th>$v_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$X_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$X_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$X_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Prefix filtering\(^2\): reading candidate sets

---

\(^1\) Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008  
BASELINES FOR FIND TOP-1

Posting list union\(^1\): reading all posting lists costs 7 values

Prefix filtering\(^2\): reading candidate sets costs 18 values

---


BASELINES FOR FIND TOP-1

Posting list union\(^1\): reading all posting lists costs 13 values

Prefix filtering\(^2\): reading candidate sets costs 7 values

---


### COST MATTERS IN DATA LAKES

<table>
<thead>
<tr>
<th>Dataset</th>
<th># of Sets</th>
<th>Max Size</th>
<th>Avg. Size</th>
<th># of Uniq. Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Data*</td>
<td>745K</td>
<td>22M</td>
<td>1,540</td>
<td>562M</td>
</tr>
<tr>
<td>WDC Web Table</td>
<td>163M</td>
<td>17K</td>
<td>10</td>
<td>184M</td>
</tr>
<tr>
<td>AOL (Query Logs)</td>
<td>10M</td>
<td>245</td>
<td>3</td>
<td>3.9M</td>
</tr>
<tr>
<td>ERON (Emails)</td>
<td>517K</td>
<td>3,162</td>
<td>135</td>
<td>1.1M</td>
</tr>
<tr>
<td>DBLP (Bibliographies)</td>
<td>100K</td>
<td>1,625</td>
<td>86</td>
<td>6,864</td>
</tr>
</tbody>
</table>

*215,393 Open Data tables from Canadian, US, and UK Open Data Portal

**Read Bottleneck** — Large number of sets and data values makes index access expensive; large posting lists and sets are expensive to read

An efficient algorithm must reduce read cost
READING SET ELIMINATES POSTING LISTS
READING SET ELIMINATES POSTING LISTS

Maximum overlap from any unseen candidates in posting lists $v_4$, $v_5$ and $v_7$ is $3 < 4$;

$|Q \cap X_4| = 4$
READING SET ELIMINATES POSTING LISTS

Maximum overlap from any unseen candidates in posting lists $v_4$, $v_5$ and $v_7$ is $3 < 4$;

$X_4$ is the top-1; eliminate posting lists $v_4$, $v_5$ and $v_7$

$|Q \cap X_4| = 4$
Current candidate sets are: \( \{X_1, X_2, X_3, X_4\} \)

Assume the current top-1 has 4 overlaps
Current candidate sets are: \( \{X_1, X_2, X_3, X_4\} \)

Assume the current top-1 has 4 overlaps

\( X_1, X_2 \) and \( X_3 \) each has only 1 more value after \( v_4 \)
Current candidate sets are: \( \{X_1, X_2, X_3, X_4\} \)

\( X_1, X_2 \) and \( X_3 \) can have at most 2 overlaps, less than the current top-1; **eliminate**

Assume the current top-1 has 4 overlaps
Current candidate sets are: 
{ }, X_4

\[ \begin{array}{cccccccc}
Q & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 \\
 X_1 & \textcolor{red}{\circ} & \textcolor{red}{\circ} & \textcolor{red}{\circ} & \textcolor{red}{\circ} & \textcolor{red}{\circ} & \textcolor{red}{\circ} & \textcolor{red}{\circ} & \textcolor{red}{\circ} \\
 X_2 & \textcolor{red}{\circ} & \textcolor{red}{\circ} & \textcolor{red}{\circ} & \textcolor{red}{\circ} & \textcolor{red}{\circ} & \textcolor{red}{\circ} & \textcolor{red}{\circ} & \textcolor{red}{\circ} \\
 X_3 & \textcolor{red}{\circ} & \textcolor{red}{\circ} & \textcolor{red}{\circ} & \textcolor{red}{\circ} & \textcolor{red}{\circ} & \textcolor{red}{\circ} & \textcolor{red}{\circ} & \textcolor{red}{\circ} \\
 X_4 & \textcolor{red}{\circ} & \textcolor{red}{\circ} & \textcolor{red}{\circ} & \textcolor{red}{\circ} & \textcolor{red}{\circ} & \textcolor{red}{\circ} & \textcolor{red}{\circ} & \textcolor{red}{\circ} \\
\end{array} \]

X_1, X_2 and X_3 each can have at most 2 overlaps, less than the current top-1; eliminate

X_1, X_2 and X_3 each has only 1 more value after v_4

Assume the current top-1 has 4 overlaps
JOSIE - JOIN SEARCH VIA INTERSECTION EST.¹

Measure “work” as total read cost of remaining posting lists and candidates

Measure “progress” as reduction in work
  • Reading posting lists reveals intersecting values – reducing work in reading candidates
  • Reading candidates improves prefix filter – reducing work in reading posting lists

Measure “price” as the read cost associated with progress

¹Erkang Zhu, Dong Deng, Fatemeh Nargeslan, Renée J. Miller, “JOSIE: Overlap Set Similarity Search for Finding Joinable Tables in Data Lakes”, SIGMOD 2019
JOSIE - JOIN SEARCH VIA INTERSECTION EST.\textsuperscript{1}

Measure “work” as total read cost of remaining posting lists and candidates

Measure “progress” as reduction in work
- Reading posting lists reveals intersecting values – reducing work in reading candidates
- Reading candidates improves prefix filter – reducing work in reading posting lists

Measure “price” as the read cost associated with progress

```plaintext
while work > 0:
    estimate progress
    choose reading posting lists or candidates
    that maximize (progress - price)
```

\textsuperscript{1} Erkang Zhu, Dong Deng, Fatemeh Nargeslan, Renée J. Miller, “JOSIE: Overlap Set Similarity Search for Finding Joinable Tables in Data Lakes”, SIGMOD 2019
JOSIE - JOIN SEARCH VIA INTERSECTION EST.¹

Measure "work" as total read cost of remaining posting lists and candidates

Measure "progress" as reduction in work
- Reading posting lists reveals intersecting values - reducing work in reading candidates
- Reading candidates improves prefix filter - reducing work in reading posting lists

Measure "price" as the read cost associated with progress

while work > 0:
    estimate progress
    choose reading posting lists or candidates
    that maximize (progress - price)

JOSIE - JOIN SEARCH VIA INTERSECTION EST.

Measure “work” as total read cost of remaining posting lists and candidates

Measure “progress” as reduction in work
- Reading posting lists reveals intersecting values – reducing work in reading candidates
- Reading candidates improves prefix filter – reducing work in reading posting lists

Measure “price” as the read cost associated with progress

```
while work > 0:
    estimate progress
    choose reading posting lists or candidates that maximize (progress - price)
```

JOSIE - PERFORMANCE

Evaluated on an Open Data lake (745K sets, 562M posting lists) and WDC Web Tables (163M sets, 184M posting lists)

• Up to 5× performance over baseline algorithms
• Up to 2× performance over LSH Ensemble (approximate), small queries (<5K) and $k < 20$
**JOSIE - PERFORMANCE**

Evaluated on an Open Data Benchmark and WDC Web Tables (1.63M sets):
- Up to 5x performance improvement
- Up to 2x performance improvement

On Open Data Benchmark
OTHER WORK

Auto-Join: Join Table by Leveraging Transformation (VLDB 17’)
- Collaboration with Yeye He and Surajit Chaudhuri @ MSR

Learning transformation without user input

<table>
<thead>
<tr>
<th>Company</th>
<th>Headquarter</th>
<th>Ticker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alphabet Inc.</td>
<td>Mountain View</td>
<td>GOOGL (NASDAQ)</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>Palo Alto</td>
<td>AAPL (NASDAQ)</td>
</tr>
<tr>
<td>Alibaba</td>
<td>Hangzhou</td>
<td>BABA (NYSE)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exchange</th>
<th>Ticker</th>
<th>Open</th>
<th>Close</th>
<th>Market Cap</th>
</tr>
</thead>
<tbody>
<tr>
<td>NASDAQ</td>
<td>GOOGL</td>
<td>1,032.50</td>
<td>1,042.32</td>
<td>766 B</td>
</tr>
<tr>
<td>NASDAQ</td>
<td>AAPL</td>
<td>171.10</td>
<td>175.12</td>
<td>892 B</td>
</tr>
<tr>
<td>NYSE</td>
<td>BABA</td>
<td>179.37</td>
<td>174.13</td>
<td>427 B</td>
</tr>
</tbody>
</table>
OTHER WORK

Table Union Search on Open Data¹ (VLDB 17’)
- Given a query table, find tables that can be “stacked” with the query table and aligning columns
- Top-K and Approximate

Optimizing Organizations for Navigating Data Lakes² (VLDB 19’ Revision)
- Generate a navigational DAG of topics for browsing the data lakes effectively
- Complementary to search

¹ Fatemeh Nargesian, Erkang Zhu, Ken Pu, Renée J. Miller, “Table Union Search on Open Data”, VLDB 2017
FUTURE WORK

1. Tables → Sets
   - Preprocessing (e.g. extract sets)

2. Sets → Search Index
   - Indexing

3. Search Index → User Interface
   - Column and Table IDs
   - Query
   - Query Set
     - Preprocessing (e.g. extract set)

4. User Interface → Table
FUTURE WORK

Other possibilities:
- Multi-sets
- Entities
- Numerical data
- Embedding vectors
FUTURE WORK

Future enhancements:
- Extension to DBMS
- Distributed index
- Improve performance

Other possibilities:
- Multi-sets
- Entities
- Numerical data
- Embedding vectors
FUTURE WORK

Future enhancements:
- Extension to DBMS
- Distributed index
- Improve performance

Potential applications:
- ML systems
- BI systems
- Citizen data science
- Data marketplace

Other possibilities:
- Multi-sets
- Entities
- Numerical data
- Embedding vectors

Preprocessing (e.g. extract sets)