Mind the Gap: Large-Scale Frequent Sequence Mining

Iris Miliaraki  Klaus Berberich  Rainer Gemulla  Spyros Zoupanos

Max Planck Institute for Informatics
Saarbrücken, Germany
Why are sequences interesting?

Google books Ngram Viewer

Graph these case-sensitive comma-separated phrases: sunny day in New York, rainy day in New York

between **1950** and **2000** from the corpus English with smoothing of **3**.

Search lots of books

---

Various applications
Why are sequences interesting?

[Google books Ngram Viewer]

Graph these case-sensitive comma-separated phrases: sunny New York, rainy New York

between 1950 and 2000 from the corpus English with smoothing of 3.

Search lots of books

[sunny New York] [rainy New York]
Sequences with gaps

• Generalization of n-grams to sequences with gaps
  – sunny [...] New York
  – rainy [...] New York

• Exposes more structure
  ✓ Central Park is the best place to be on a sunny day in New York.
  ✓ It was a sunny, beautiful New York City afternoon.
More applications....

- **Text analysis** (e.g., linguistics or sociology)
- **Language modeling** (e.g., query completion)
- **Information extraction** (e.g., relation extraction)
- Also: web usage mining, spam detection, ...
Challenges

Huge collections of sequences

Computationally intensive problem
- $O(n^2)$ n-grams for sequence $S$ where $|S| = n$
- $O(2^n)$ subsequences for sequence $S$ where $|S| = n$ and gap $> n$

Sequences with small support can be interesting

Potentially many output patterns

How can we perform frequent sequence mining at such large scales?
Outline

- Motivation & challenges
- **Problem statement**
- The MG-FSM algorithm
- Experimental Evaluation
- Conclusion
Gap-constrained frequent sequence mining

**Input:** Sequence database

**Output:** Frequent subsequences that

- Occur in at least $\sigma$ sequences (support threshold)
- Have length at most $\lambda$ (length threshold)
- Have gap at most $\gamma$ between consecutive items (gap threshold)

Central Park is the best place to be on a sunny day in New York.
Monday was a sunny day in New York.
It was a sunny, beautiful New York City afternoon.

Frequent n-gram sunny day in New York for $\sigma = 2$, $\gamma = 0$, $\lambda = 5$
Gap-constrained frequent sequence mining

**Input:** Sequence database

**Output:** Frequent subsequences that
- Occur in at least $\sigma$ sequences (support threshold)
- Have length at most $\lambda$ (length threshold)
- Have gap at most $\gamma$ between consecutive items (gap threshold)

---

Central Park is the best place to be on a sunny day in New York.
Monday was a sunny day in New York.
It was a sunny, beautiful New York City afternoon.

Frequent n-gram: sunny day in New York
for $\sigma = 2$, $\gamma = 0$, $\lambda = 5$

Frequent n-gram: New York
for $\sigma = 3$, $\gamma = 0$, $\lambda = 2$
Gap-constrained frequent sequence mining

**Input:** Sequence database

**Output:** Frequent subsequences that
- Occur in at least $\sigma$ sequences (support threshold)
- Have length at most $\lambda$ (length threshold)
- Have gap at most $\gamma$ between consecutive items (gap threshold)

Central Park is the best place to be on a sunny day in New York.
Monday was a sunny day in New York.
It was a sunny, beautiful New York City afternoon.

Frequent n-gram: sunny day in New York for $\sigma = 2$, $\gamma = 0$, $\lambda = 5$
Frequent n-gram: New York for $\sigma = 3$, $\gamma = 0$, $\lambda = 2$
Frequent subsequence: sunny New York for $\sigma = 3$, $\gamma \geq 2$, $\lambda = 3$
Outline

- Motivation & challenges
- Problem statement
- The MG-FSM algorithm
- Experimental Evaluation
- Conclusion
Parallel frequent sequence mining

1. Divide data into potentially overlapping partitions
2. Mine each partition
3. Filter and combine results
Using item-based partitioning

1. Order items by desc. frequency $a > ... > k$
2. Partition by item $a, b, ...$ (called pivot item)
3. Mine each partition
4. Filter: no less-frequent item

Disjoint subsequence sets computed in-parallel and independently

---

Mind the Gap: Large-Scale Frequent Sequence Mining
Example: Naive partitioning

Support $\sigma = 2$
Max. gap $\gamma = 1$
Max. length $\lambda = 3$
Example: Naive partitioning

Support $\sigma=2$
Max. gap $\gamma=1$
Max. length $\lambda=3$
Mind the Gap: Large-Scale Frequent Sequence Mining

Example: Naive partitioning

- High communication cost
- Redundant computation cost

Support $\sigma=2$

Max. gap $\gamma=1$

Max. length $\lambda=3$

A: 6
B: 4
C: 4

A, B, C, D, AA, AB, AC, AD, BA, BC, CA

A, B, C, AA, AB, AC, BA, BC

A, B, C, AD

A, B, AD

A, D, AD

A, D
Improving the partitioning

Traditional approach

- Derive a partitioning rule ("projection")
- Prove correctness of the partition rule

MG-FSM approach

- Use *any* partitioning satisfying correctness
- Rewrite the input sequences $\rightarrow$
  ensuring each $w$-partition generates the set of *pivot sequences*
  for $w$
Which is the optimal partition?

Max. gap $\gamma = 0$
Max. length $\lambda = 2$
pivot C

Many short sequences? Few long sequences?
Optimal partition not clear!

Aim for a “good” partition using inexpensive rewrites

Trade-off cost & gain
Rewriting partitions

Partition C

A C B
D A C B D
D A C B D D B C A
B C A D D B D
A D D C D

Frequent sequences with C but not D

γ = 1
λ = 3

1. Replace irrelevant items (i.e., less frequent) by special blank symbol (\_)

Mind the Gap: Large-Scale Frequent Sequence Mining
### Rewriting partitions

<table>
<thead>
<tr>
<th>Partition C</th>
<th>frequent sequences with C but not D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A C B</td>
<td>γ = 1</td>
</tr>
<tr>
<td>D A C B D</td>
<td>λ = 3</td>
</tr>
<tr>
<td>D A C B D D B C A</td>
<td></td>
</tr>
<tr>
<td>B C A D D B D</td>
<td></td>
</tr>
<tr>
<td>A D D C D</td>
<td></td>
</tr>
</tbody>
</table>

1. Replace irrelevant items (i.e., less frequent) by special blank symbol (_).
Rewriting partitions

Partition C

A C B
_ A C B_
_ A C B _ _ B C A
B C A _ _ B_
A _ _ C _

frequent sequences with C but not D

γ = 1
λ = 3

1. Replace irrelevant items (i.e., less frequent) by special blank symbol ( _)
Rewriting partitions

Partition C → frequent sequences with C but not D

\[\begin{array}{ccc}
A & C & B \\
_ & A & C & B \\
_ & A & C & B & _ & B & C & A \\
B & C & A & _ & _ & B & _ \\
A & _ & _ & C & _ \\
\end{array}\]

\(\gamma = 1\)
\(\lambda = 3\)

1. Replace irrelevant items (i.e., less frequent) by special blank symbol (\(\_\))

2. Drop irrelevant sequences (i.e., cannot generate any pivot sequence)
Rewriting partitions

Partition C

\[
\begin{align*}
A & \; C & \; B \\
\_ & A & C & B \\
\_ & A & C & B & \_ & B & C & A \\
B & C & A & \_ & \_ & B & \_ \\
A & \_ & \_ & C & \_ \\
\end{align*}
\]

\textit{frequent sequences with C but not D}

\(\gamma = 1\)
\(\lambda = 3\)

1. Replace irrelevant items (i.e., less frequent) by special blank symbol (\_)
2. Drop irrelevant sequences (i.e., cannot generate any pivot sequence)
Rewriting partitions

Partition C

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C</td>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>_</td>
<td>A</td>
<td>C</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>_</td>
<td>A</td>
<td>C</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>C</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>A</td>
<td></td>
<td>B</td>
</tr>
</tbody>
</table>

\[ \text{frequent sequences with C but not D} \]

\[ \gamma = 1 \]
\[ \lambda = 3 \]

1. Replace irrelevant items (i.e., less frequent) by special blank symbol (\( \_ \))
2. Drop irrelevant sequences (i.e., cannot generate any pivot sequence)
3. Remove all unreachable items (provably correct)
Rewriting partitions

1. Replace irrelevant items (i.e., less frequent) by special blank symbol (\_)
2. Drop irrelevant sequences (i.e., cannot generate any pivot sequence)
3. Remove all unreachable items (provably correct)

Partition C

\[
\begin{array}{ccc}
A & C & B \\
\_ & A & C & B \\
\_ & A & C & B & \_ & B & C & A \\
B & C & A & \_
\end{array}
\]

\(\gamma = 1\)
\(\lambda = 3\)

frequent sequences with C but not D
Rewriting partitions

1. Replace irrelevant items (i.e., less frequent) by special blank symbol (\_)
2. Drop irrelevant sequences (i.e., cannot generate any pivot sequence)
3. Remove all unreachable items (provably correct)
4. Remove trailing and leading blanks

Partition C

\[\begin{array}{cccc}
A & C & B \\
\_ & A & C & B \_ \\
\_ & A & C & B & \_ & \_ & B & C & A \\
B & C & A \_ \\
\end{array}\]

\textit{frequent sequences with C but not D}

\[\gamma = 1, \quad \lambda = 3\]
Rewriting partitions

Partition C

A C B
A C B
A C B _ _ B C A
B C A

frequent sequences with C but not D

γ = 1
λ = 3

1. Replace irrelevant items (i.e., less frequent) by special blank symbol (_)
2. Drop irrelevant sequences (i.e., cannot generate any pivot sequence)
3. Remove all unreachable items (provably correct)
4. Remove trailing and leading blanks
Rewriting partitions

Partition C

A C B
A C B
A C B _ _ B C A
B C A

frequent sequences with C but not D

γ = 1
λ = 3

1. Replace irrelevant items (i.e., less frequent) by special blank symbol (\_)
2. Drop irrelevant sequences (i.e., cannot generate any pivot sequence)
3. Remove all unreachable items (provably correct)
4. Remove trailing and leading blanks
5. Break up sequence at split points (i.e., sequences of \(γ+1\) blanks)
Rewriting partitions

Partition C → frequent sequences with C but not D

γ = 1
λ = 3

1. Replace irrelevant items (i.e., less frequent) by special blank symbol (_)
2. Drop irrelevant sequences (i.e., cannot generate any pivot sequence)
3. Remove all unreachable items (provably correct)
4. Remove trailing and leading blanks
5. Break up sequence at split points (i.e., sequences of γ+1 blanks)
Rewriting partitions

1. Replace irrelevant items (i.e., less frequent) by special blank symbol (\_)
2. Drop irrelevant sequences (i.e., cannot generate any pivot sequence)
3. Remove all unreachable items (provably correct)
4. Remove trailing and leading blanks
5. Break up sequence at split points (i.e., sequences of $\gamma + 1$ blanks)
6. Aggregate repeated subsequences
Rewriting partitions

Partition C

<table>
<thead>
<tr>
<th>A C B</th>
<th>B C A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \text{γ} = 1 \quad \text{λ} = 3 \]

\[ \text{frequent sequences with C but not D} \]

1. Replace irrelevant items (i.e., less frequent) by special blank symbol (\( _{\ } \))
2. Drop irrelevant sequences (i.e., cannot generate any pivot sequence)
3. Remove all unreachable items (provably correct)
4. Remove trailing and leading blanks
5. Break up sequence at split points (i.e., sequences of \( \gamma + 1 \) blanks)
6. Aggregate repeated subsequences

Mind the Gap: Large-Scale Frequent Sequence Mining
Revisiting example: Naive partitioning

A B A B
A B C C B
A A
A D
C A D
B A C A
B C
A:6
B:4
C:4
D:2
Support $\sigma=2$
Max. gap $\gamma=1$
Max. length $\lambda=3$

Mind the Gap: Large-Scale Frequent Sequence Mining
Revisiting example: MG-FSM partitioning

A B A B
A B C C B
A
A D
C A D
B A C A
B C

A:6
B:4
C:4
D:2

Support σ=2
Max. gap γ=1
Max. length λ=3

A, D, AD

A B A B
A B
B A A

A A
with A but not B, C, D

A B A B
A B
B A A

AA, AB, BA
with B but not C, D...

A B C C B
C A
B A C A
B C

AC, BC, CA
with C but not D

A D
C A D

AD
with D
Revisiting example: MG-FSM partitioning

Mind the Gap: Large-Scale Frequent Sequence Mining
Outline

- Motivation & challenges
- Problem statement
- The MG-FSM algorithm
- Experimental Evaluation
- Conclusion
Experimental evaluation: Setup

• Algorithms
  – MG-FSM
  – Naive algorithm for MapReduce
  – Suffix-σ (state-of-the-art n-gram miner)

• Setting
  – 10-machine local cluster
  – 10 GBit/64GB of main memory/eight 2TB SAS 7200 RPM hard disks/2 Intel Xeon E5-2640 6-core CPUs
  – Cloudera cdh3u0 distribution of Hadoop 0.20.2.
## Experimental evaluation: Datasets

<table>
<thead>
<tr>
<th></th>
<th>ClueWeb</th>
<th>New York Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average length</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>Maximum length</td>
<td>20,993</td>
<td>21,174</td>
</tr>
<tr>
<td>Total sequences</td>
<td>1,135,036,279</td>
<td>531,375,076</td>
</tr>
<tr>
<td>Total items</td>
<td>21,565,723,440</td>
<td>1,051,435,745</td>
</tr>
<tr>
<td>Distinct items</td>
<td>7,361,754</td>
<td>1,577,233</td>
</tr>
<tr>
<td>Total bytes</td>
<td>66,181,963,922</td>
<td>3,087,605,146</td>
</tr>
</tbody>
</table>
n-gram mining ($\gamma=0$)

✓ Orders of magnitude faster than Naive
✓ Competitive to state-of-the-art n-gram miners

![Bar Chart]

- Naive
- Suffix-$\sigma$
- MG-FSM

Mind the Gap: Large-Scale Frequent Sequence Mining
MG-FSM partition optimizations (time)

NYT (100,1,5)

✓ 50x faster than Naive which finished after 225 mins
Strong scalability

\[ \sigma = 1000, \gamma = 1, \lambda = 5 \]

50% ClueWeb

Linear scalability as we increase machines for both map & reduce tasks
Outline

- Motivation & challenges
- Problem statement
- The MG-FSM algorithm
- Experimental Evaluation
- Conclusion
Summary & Contributions

- MG-FSM mines frequent sequences with gap constraints
- Uses item-based partitioning \(\rightarrow\) partitions can be mined independently and in parallel using any FSM algorithm
- Instead of “optimal” partitioning, MG-FSM uses efficient, inexpensive rewrites that ensure correctness
- Fast, low communication cost, scalable

Questions?