Scalable Frequent Sequence Mining With Flexible Subsequence Constraints

Alexander Renz Wieland \textsuperscript{1} \hspace{1cm} Matthias Bertsch \textsuperscript{2} \hspace{1cm} Rainer Gemulla \textsuperscript{2}

\textsuperscript{1}Technische Universität Berlin

\textsuperscript{2}Universität Mannheim

ICDE 2019, Macau, China
April 11\textsuperscript{th}, 2019
Frequent Sequence Mining (FSM)

Fundamental task in data mining
- Data modeled as sequences of items or events
- Often items are arranged in a hierarchy
- Goal is to discover frequent subsequences

Example (market-basket data)
- Sequence = purchases of customer over time
- Item = product + product hierarchy
- Example subsequence = DSLR Camera → Tripod → Flash

Applications
- Natural language processing
- Information extraction
- Web usage analysis
- . . .
Challenge: Flexibility

- Unconstrained FSM outputs a multitude of frequent subsequences
  
a bell (302392),
become president (234311),
**graduated from (3962)**,
why so many of us (234),
of the (220125),
going to (12897),

had never used (23202),
PER be professor (1582),
large enough to be (12083),
who VERB also (22 223),
**lives in (4322)**,
great artist (2394),
...

- Typically, only few of them are interesting to a specific application
  - E.g., only relational phrases between entities are of interest

- Flexible methods (that can be tailored to applications) are essential
Goal: flexible and scalable FSM

- Common approach: flexible subsequence constraints

- Problem: existing FSM algorithms are flexible or scalable

- Our paper: flexible and scalable
Outline

1. Frequent Sequence Mining

2. Flexibility

3. Scalability

4. Conclusion
Flexible FSM with DESQ

- We adopt the unified FSM framework **DESQ** [ICDM ’16, TODS ’19]
  - Applications can describe flexible subsequences constraints in an intuitive, declarative way
  - Alleviates need for customized mining algorithms

- Provides **pattern expression language** to specify subsequence constraints
  - Syntax like regular expressions
  - Supports captures groups and hierarchies
Example pattern expressions for applications

1. **Noun modified by adjective or noun**
   - big country (110), research scientist (473)

2. **Relational phrase between entities**
   - is being advised by (15), has coached (10)

3. **Products bought after a digital camera**
   - Camera Lenses, Tripods & Monopods (11), Camera Batteries, SD & SDHC Cards (12)

4. **Amino acid sequences that match**
   - [S | T].[R | T]
   - SLR(103,093), TAK(102941)
Example pattern expressions for traditional constraints

1. 3-grams  
2. 3-, 4-, and 5-grams  
3. skip 3-grams with gap 1  
4. All subsequences  
5. length 3–5 subsequences  
6. bounded gap of 0–3  
7. serial episodes of length 3, window 5  
8. generalized 5-grams  
9. subsequences matching regex \([a|b] \, c^* \, d\)  
10. ...
Outline

1. Frequent Sequence Mining

2. Flexibility

3. Scalability
   3.1 General framework
   3.2 Communicate inputs
   3.3 Communicate candidates
   3.4 Experimental study

4. Conclusion
A general framework for distributed FSM

- Bulk synchronous parallel with 1 round of communication

1. Local preprocessing (map)
2. Communication (shuffle)
3. Local mining (reduce)

- Item-based partitioning [SIGMOD '00, PPoPP '07, SIGMOD '13]

Input sequence: acdcb
Candidate subsequences:
- Relevant for partition c: acdcb, acdb, acb
- Relevant for partition a: adcb, accb
- (not relevant for partitions b, d: adb, ab)

- Key challenges
  - How to distribute computation
  - What to communicate
Communicate inputs

- Send each input sequence to all partitions to which it can contribute

1. Determine partitions, rewrite input sequences
2. Send rewritten input sequences
3. Run local FSM algorithm

- Often sufficient to send parts of the input sequence

- Example: if e’s not relevant for mining task, don’t send them

A. Renz-Wieland, M. Bertsch, R. Gemulla
Scalable Frequent Sequence Mining With Flexible Subsequence Constraints
Communicate candidates

- Send each candidate subsequence to its corresponding partition

1. Generate and compress candidates
2. Send compressed candidates
3. Count candidates

Important optimization: compress candidates

\[ \text{acdcdb, acdb, acb, adcb, accb} \]

\[ \text{acdcdb, acdb, acb, adcb, accb} \]

\[ \text{acdcdb, acdb, acb, adcb, accb} \]
Experimental study: key results

- Up to 50x faster than naïve approaches, up to 100x less communication

(a) New York Times data

(b) Amazon Review data

- Sending candidates is up to 5x faster for selective constraints

- 1-4x generalization overhead over specialized, less general approaches

- Both approaches scale nearly linearly with number of input sequences
Outline

1. Frequent Sequence Mining
2. Flexibility
3. Scalability
4. Conclusion
Conclusion

- Existing algorithms: **flexible or scalable**. Ours: both

- Adopt DESQ: a framework to tailor FSM to applications

- Distributed mining via item-based partitioning
  1. Communicate inputs
  2. Communicate candidates

- Available as open source Apache Spark library, link at https://github.com/rgemulla/desq/tree/distributed


K. Beedkar and R. Gemulla. DESQ: Frequent sequence mining with subsequence constraints. ICDM '16.


J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. SIGMOD '00.

I. Miliaraki et al. Mind the gap: Large-scale frequent sequence mining. SIGMOD '13.