LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Institut für Informatik

Moment:
Maintaining Closed Frequent Itemsets over a Stream Sliding Window

Yun Chi, Haixun Wang, Philip S. Yu, Richard R. Muntz

Hauptseminar Data Science - SS 2016

Team D:
Julian Kolarz
Florian Wirth
Michael Spitzer

Übersicht

1. Wdh. Frequent Item Sets + Problemstellung
2. Moment Algorithmus + CET
3. Vor-/Nachteile & Performance
4. Anwendungsideen

Frequent Item Sets

Items $I = \{i_1, \ldots, i_n\}$ eine Menge von Literalen

(z.B. alle Waren in einem Supermarkt)

Itemset X: Menge von Items $X \subseteq I$

(z.B. {A, C, D} oder {Butter, Eier, Milch})

k-Itemset: ein Itemset der Länge *k*

{A, B, C} ist ein 3-Itemset

{B, D} ist ein 2-Itemset

Frequent Item Sets

Transaktion T = (tid, X_T) z.B. (2, {A,C,D})

Datenbank DB: Menge von Transaktionen T

- Frequent Item Set
- Closed Frequent Item Set
- Maximal Frequent Item Set

Frequent Item Sets

Beispiel: |E| = {A,B,C,D} s = 0,5

DB = {(1,{C,D}),
(2,{A,B}),
(3,{A,B,C}),
(4,{A,B,C})}

Frequent Item Sets:

F = {(A,3),(B,3),(C,3),(AB,3),(AC,2),(BC,2),(ABC,2)}

Closed Frequent Item Sets:

C = {(C,3),(AB,3),(ABC,2)}

Maximal Frequent Itemsets:

M = {(ABC,2)}

Problem:

Supermarktwaren (|E|): 1000+

Normaler Einkauf (Transaktion T): 10-20 Items

- Alle auftretenden Itemsets in einem Baum zu speichern sehr **speicherintensiv**
- Update sehr **zeitaufwendig** / **nicht** in Echtzeit möglich

Gewünschte Eigenschaften an die Speicherstruktur:

- Speicher-effizient
- Update-effizient

- Moment Overview

- Data-Stream + Sliding Window of size n
- Uses a data structure (CET) to monitor a dynamically selected small set of itemsets
- „What are the current closed frequent itemsets?“

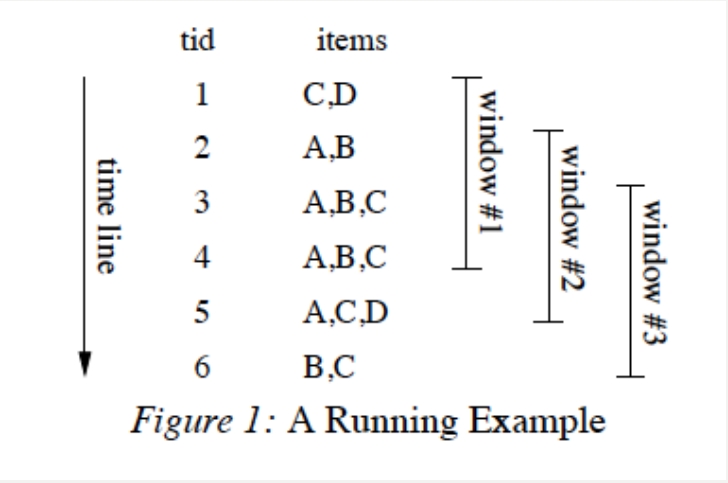
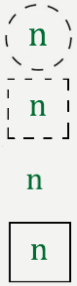


Figure 1: A Running Example

Moment: The Closed Enumeration Tree (CET)

- In-memory data structure
- For maintaining a dynamically selected set of itemsets, which contain
  1. Closed frequent itemsets
  2. Itemsets that form a boundary

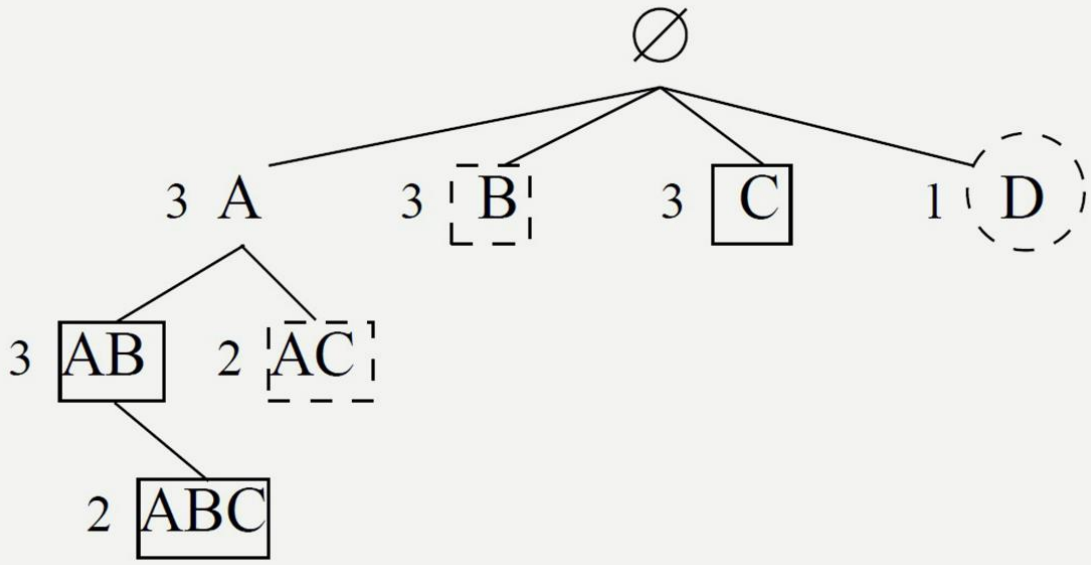
- Node Types
  - infrequent gateway nodes
  - unpromising gateway nodes
  - intermediate nodes
  - closed nodes



CET: Example

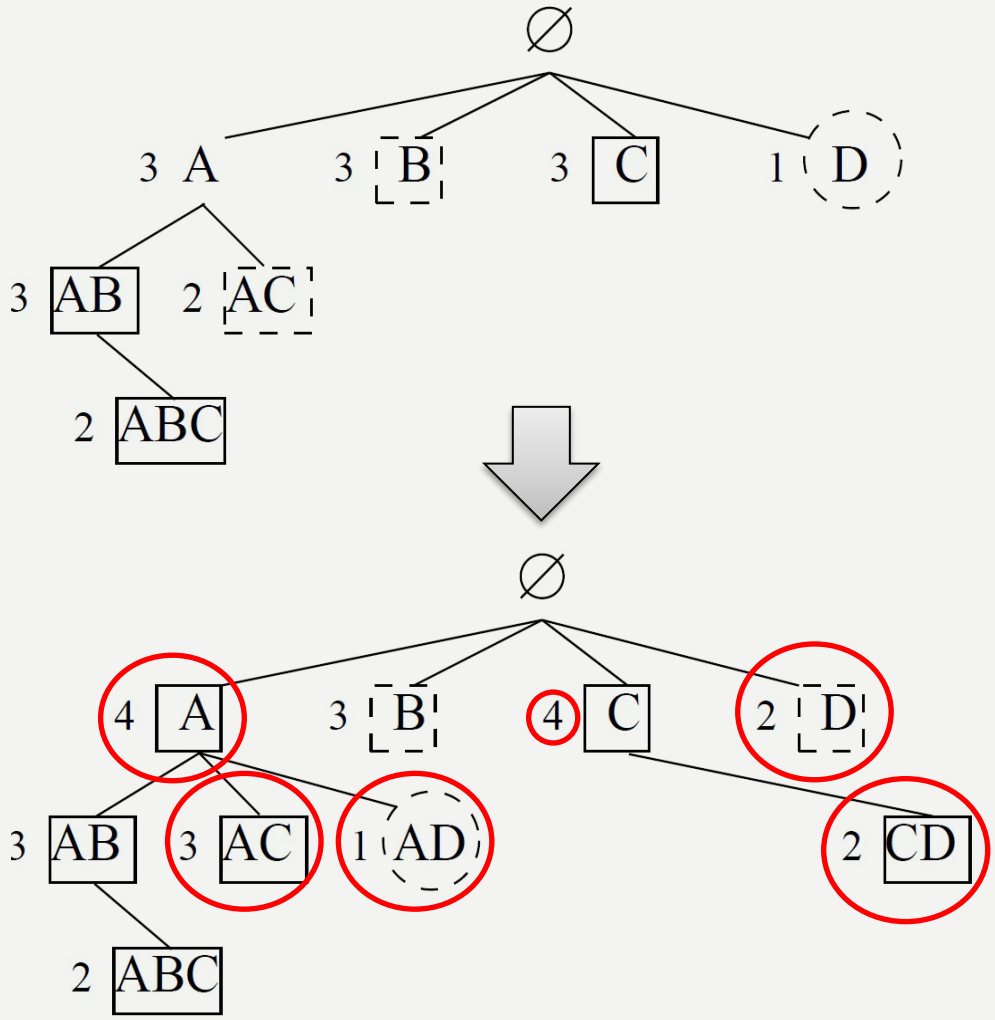
set of items	$\sum =$	$\{A, B, C, D\}$
transactions	$\mathrm{D} =$	$\{CD, AB, ABC, ABC, ACD\}$
window size	$\mathrm{N} =$	4
minimum support	$\mathrm{s} =$	2

tid	items	
1	C,D	window #1
2	A,B	
3	A,B,C	
4	A,B,C	



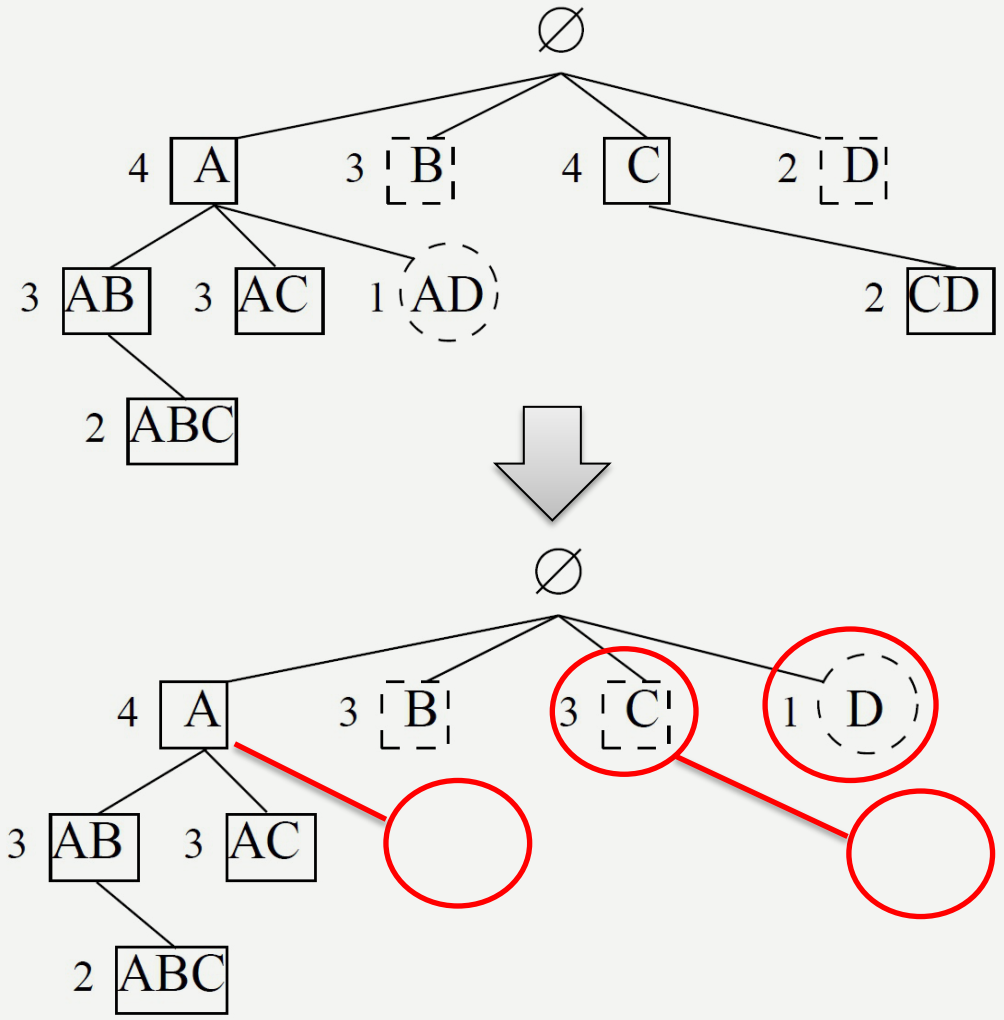
CET: Adding a Transaction

tid	items
1	C,D
2	A,B
3	A,B,C
4	A,B,C
5	A,C,D



CET: Deleting a Transaction

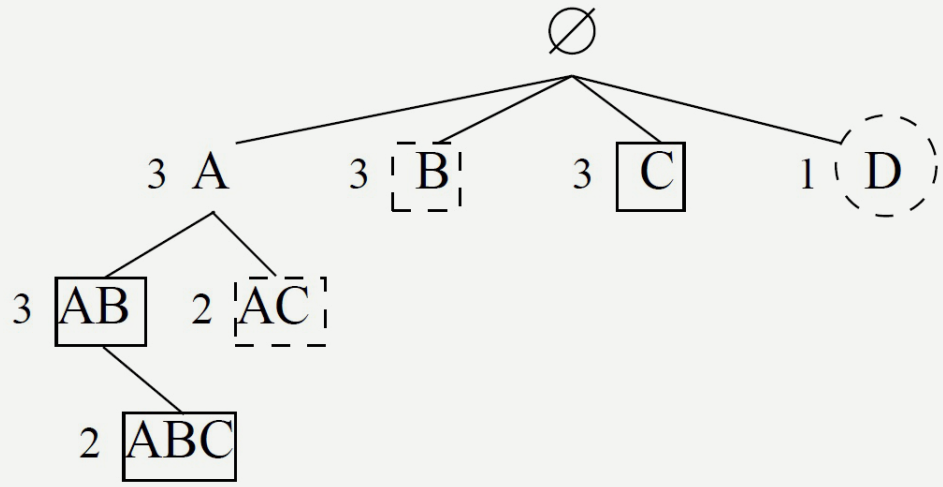
tid	items
1	C,D
2	A,B
3	A,B,C
4	A,B,C
5	A,C,D



CET: Update Summary

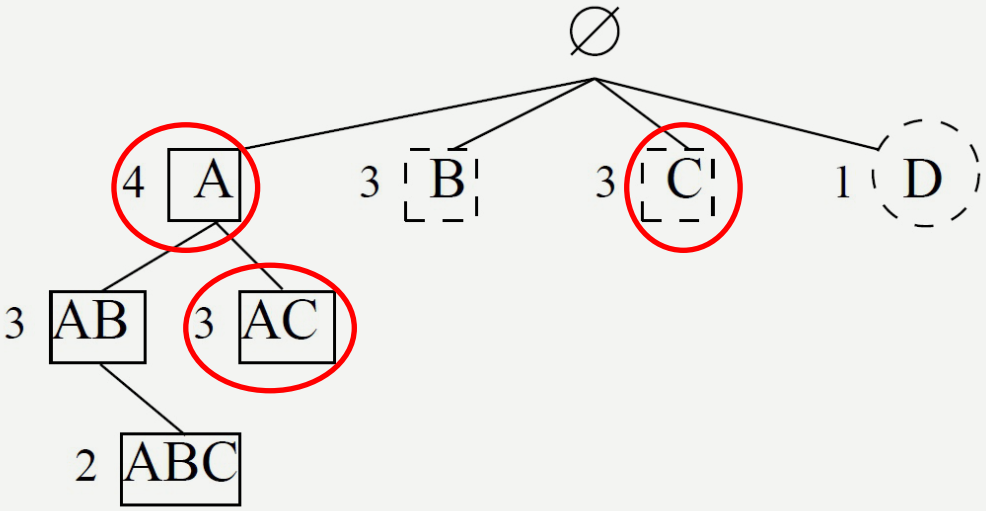
tid	items
1	C,D
2	A,B
3	A,B,C
4	A,B,C

window #1



tid	items
2	A,B
3	A,B,C
4	A,B,C
5	A,C,D

window #2



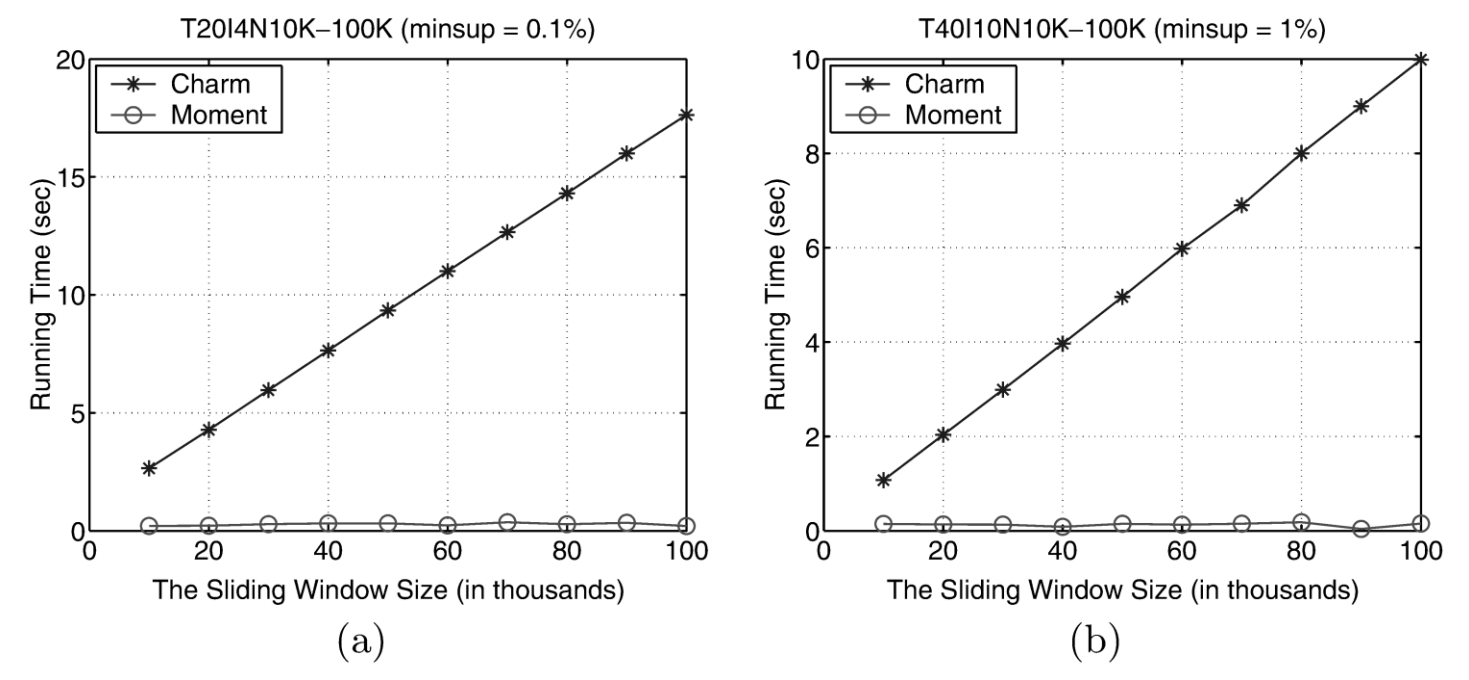
Discussion

- ‘Exlpore‘-algorithm is most time consuming operation
- The CET will maintain the current Closed Frequent Itemsets at any time
- The size for the sliding-window is flexible
- So far the algorithm only handles one transaction in one update
  -> Can be modified easily

Results

Comparison to other algorithms

Charm:



Comparison to other algorithms

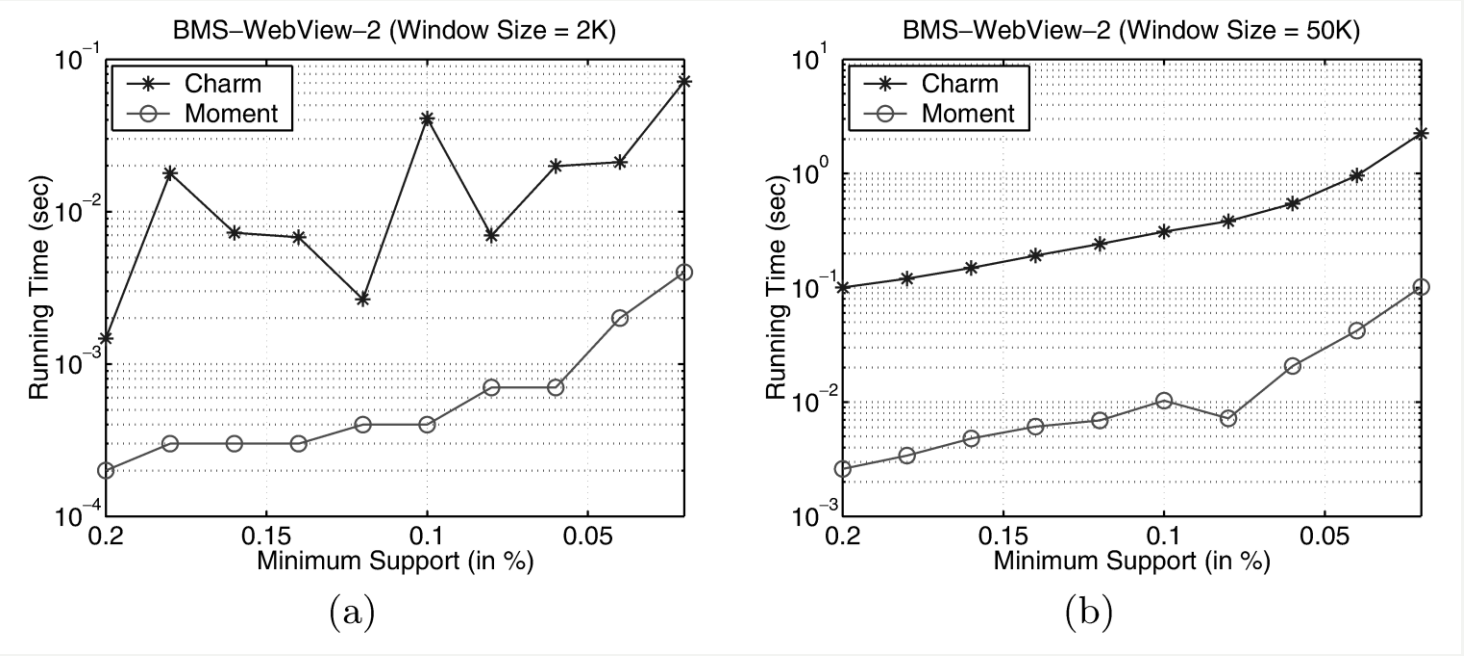
ZIGZAG:

Running-time (in seconds) comparison between Moment and ZIGZAG

Size of batch update	WCup		WPortal	
	Moment	ZIGZAG	Moment	ZIGZAG
	Sliding Window Size = 2 K			
1	**0.0267**	**1.24**	**0.000230**	**0.0617**
10	0.267	1.22	0.00230	0.0618
100	2.67	1.10	0.0230	0.0623
	Sliding Window Size = 100 K			
1	**0.0980**	**1.71**	**0.00139**	**0.229**
10	0.980	1.71	0.0139	0.227
100	9.80	1.71	0.139	0.228

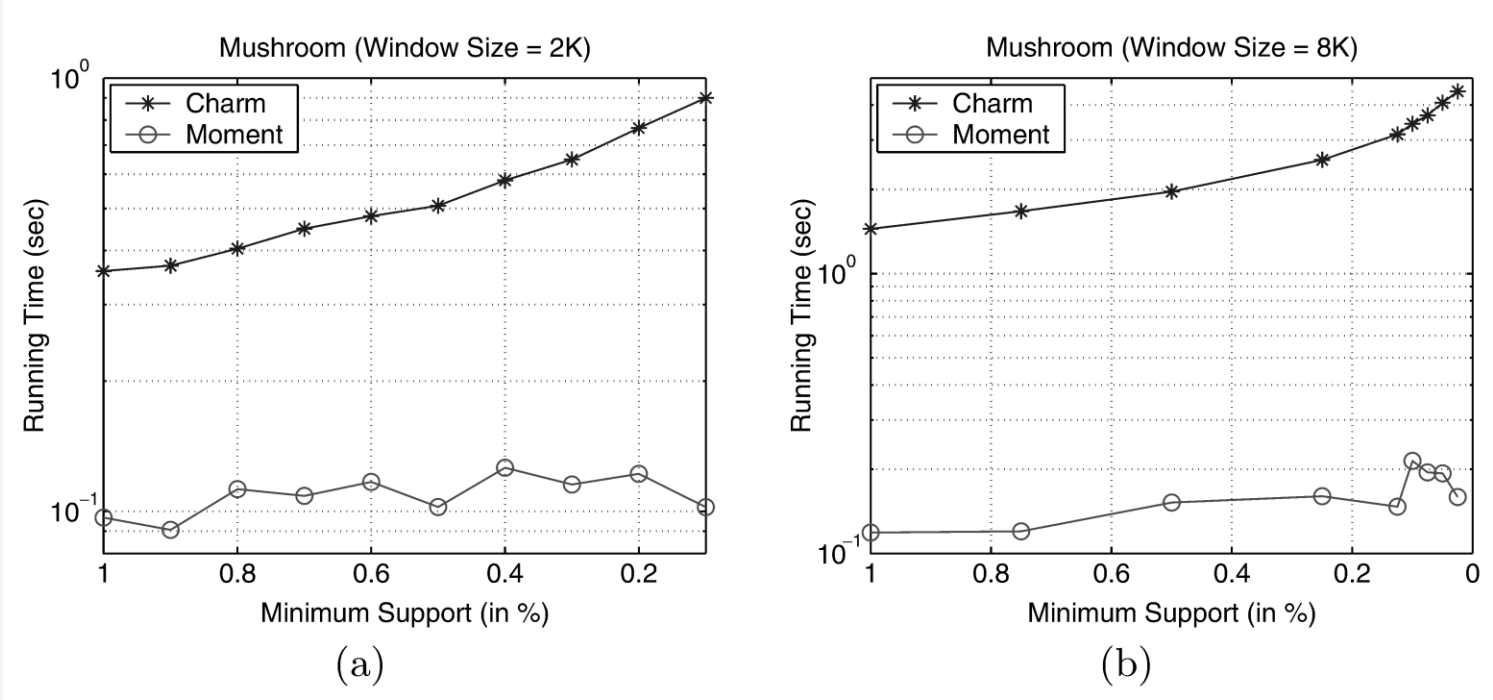
Results

Comparison for different data sets:



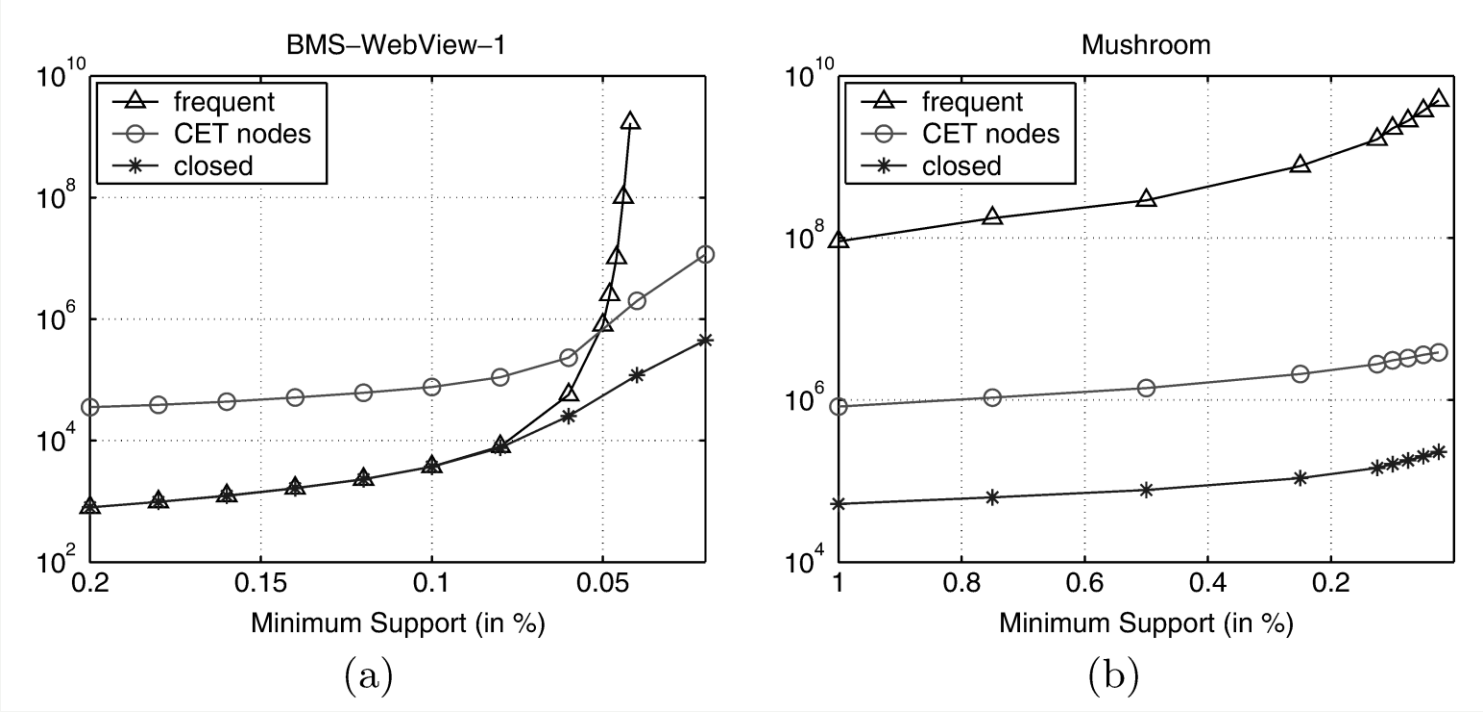
Results

Comparison for different data sets:



Results

Comparison for different Minimum Supports:



Hashtags als Frequent Itemsets:

Verfahren:

Analyse der Frequent Itemsets z.B. über ein Sliding Window des letzten Monats.

→ Welche Hashtags tauchen zur Zeit gemeinsam auf?

Anwendung:

Trendanalyse: Welche Themen/Personen/Kampagnen werden miteinander assoziiert?

`Beispiel: #Böhmermann, #Erdogan, #Schmähkritik`

Variante:

Gezielt Hashtag wählen und nur Sets mit diesem suchen → Viele Tweets uninteressant

Textteile als Frequent Itemsets:

Umsetzung:

Textprocessing der Tweets, danach Analyse der Frequent Itemsets über ein Fenster.

→ Welche Begriffe (ggf in Kombination mit Hashtags) tauchen zur Zeit gemeinsam auf?

Anwendung:

Trendanalyse: Ggf präziser als nur über Hashtags, aufwendiger.

Hürden:

Textprocessing

Von Textteilen auf Hashtags schließen:

Umsetzung:

Textprocessing der Tweets, danach Analyse Häufig gemeinsam auftauchender Textteil/Tag-Kombinationen.
→ Welche Textteile tauchen mit welchen Hashtags zusammen auf?

Anwendung:

Tag Prediction: Welche Tags passen zum eingegebenen Text?

Hürden:

Textprocessing, Verknüpfung von Textteil-Sets zu Hashtag-Sets

Varianten:

Von Tags auf Inhalt schließen

Hastags/Textteile und Geodaten verbinden:

Umsetzung:

Nur Tweets mit Geodaten, Häufigkeit einzelner Hashtags nach Ort aufschlüsseln

→ Häufen sich Begriffe/Tags an bestimmten Orten?

Anwendung:

Ortsbezogene Trends erkennen.

`Beispiel: (Texas, #VoteTrump), (Florida, #VoteHillary)`

Hürden:

Ausreichende Menge an Daten

Follower mit Hashtags/Textteilen verbinden:

Umsetzung:

Follower einer Person finden, dann häufige Hashtags/Worte in deren Tweets analysieren

→ Wessen Follower sprechen worüber?

Anwendung:

Fangemeinde/Anhängerschaft analysieren

Hürden:

Ausreichende Menge an Daten, Persönlichkeitsrechte

Variante:

Statt Followern Friends analysieren: Worüber schreiben Menschen, denen ich folge?
Ggf Rückschluss auf mich selbst möglich → Ähnliche Hürden

Follower mit Geodaten verbinden:

Umsetzung:

Follower einer Person finden und diese (wo vorhanden) auf Geodaten mappen

→ Wer hat wo Follower?

Anwendung:

Fangemeinde/Anhängerschaft analysieren

`Beispiel: Lokale Berühmtheit vs. internationale Bekanntheit?`

Hürden:

Ausreichende Menge an Daten, Persönlichkeitsrechte

Variante:

Statt Followern Friends analysieren: Wo leben Menschen, denen ich folge?
Ggf Rückschluss auf mich selbst möglich → Ähnliche Hürden

Vielen Dank für Ihre Aufmerksamkeit