POLITECHNIKA WROCŁAWSKA
WYDZIAŁ INFORMATYKI I TELEKOMUNIKACJI

# METODY ANALIZY I EKSPLORACJI DANYCH

## Wykład 11 - Big data. Dane strumieniowe. Odkrywanie wzorców sekwencji.


DR INŻ. AGATA MIGALSKA

01

BIG DATA

02

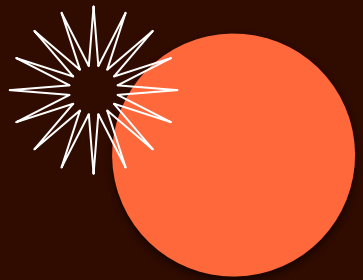
ANALIZA DANYCH STRUMIENIOWYCH

03

ODKRYWANIE WZORCÓW SEKWENCJI

04

PRZYKŁAD W BIG DATA

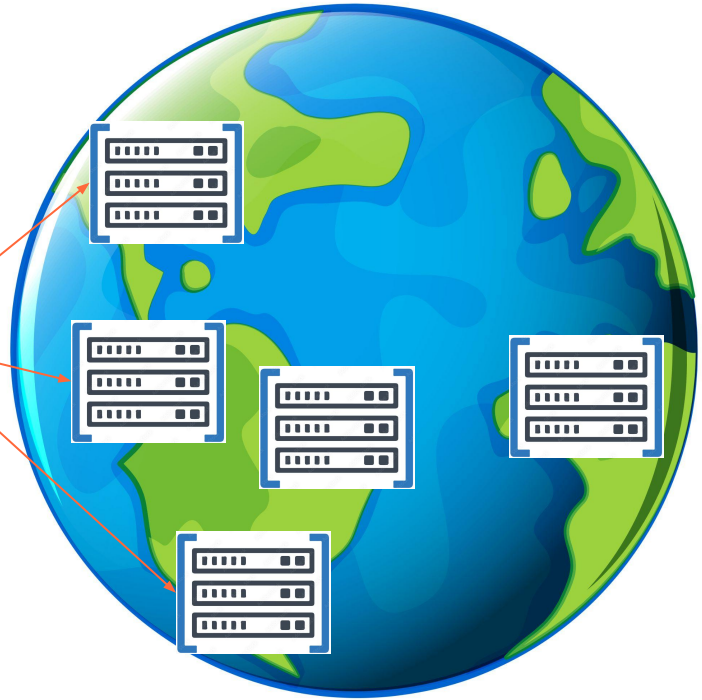


01

# BIG DATA & MAP-REDUCE

# JAK OPTYMALNIE ROZDZIELIĆ ZADANIA?

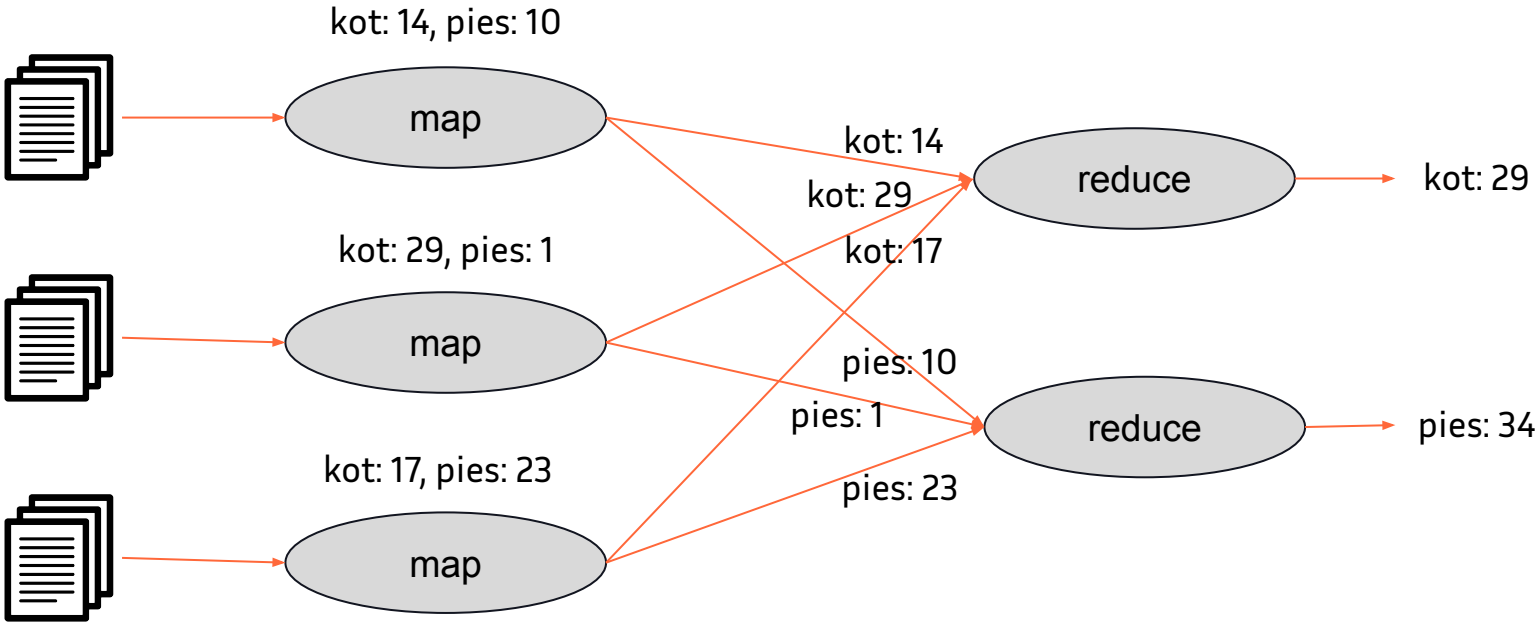
CENTRA DANYCH



ZADANIA:


Źródło grafiki: https://pl.freepik.com/darmowe-wektory/na-bialym-tle-ziemia-na-bialym-tle_13774164.htm

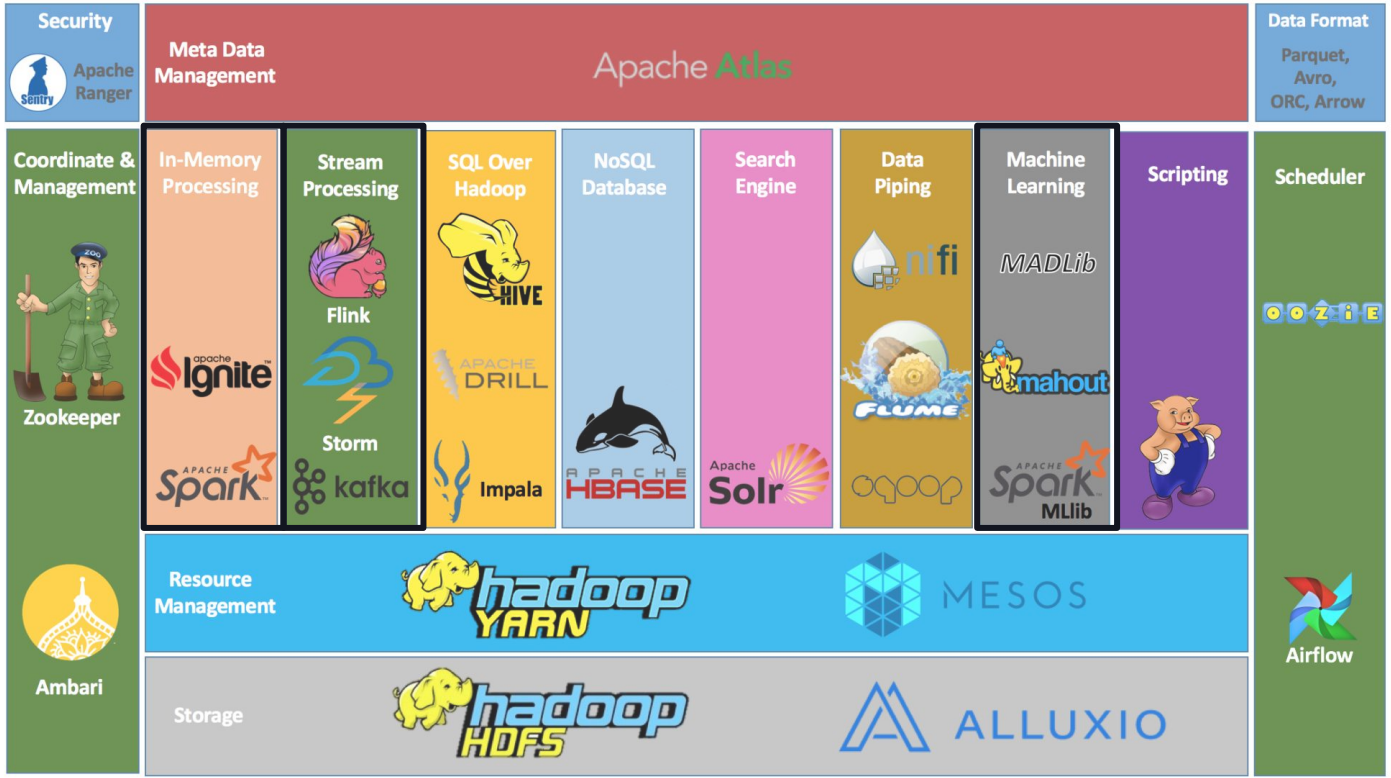
# JAK ZLICZYĆ LICZBĘ SŁÓW W 100,000 DOKUMENTÓW?



# PARADYGMAT MAP-REDUCE

- Koncepcja efektywnego zrównoleglenia obliczeń stworzona w 2003 roku przez Google.
- Celem modelu jest umożliwienia przetwarzania dużych zbiorów danych na masową skalę w równoległy sposób.
- Dane wejściowe są podzielone na niezależne fragmenty. Każdy fragment jest przetwarzany równolegle w węzłach w klastrze. W ogólności, system MapReduce składa się z dwóch komponentów / funkcji:
    - Maper: używa danych wejściowych, analizuje je (zwykle z operacjami filtrowania i sortowania) i emituje krotki (pary klucz-wartość).
    - Reduktor: zużywa krotki emitowane przez narzędzie Maper i wykonuje operację podsumowania, która tworzy mniejszy, połączony wynik z danych mapowania.
- Pierwsza implementacja - Google w C++ (2003).
- Pierwsza “zewnętrzna” implementacja - Apache Hadoop (2006)

# APACHE HADOOP ECOSYSTEM

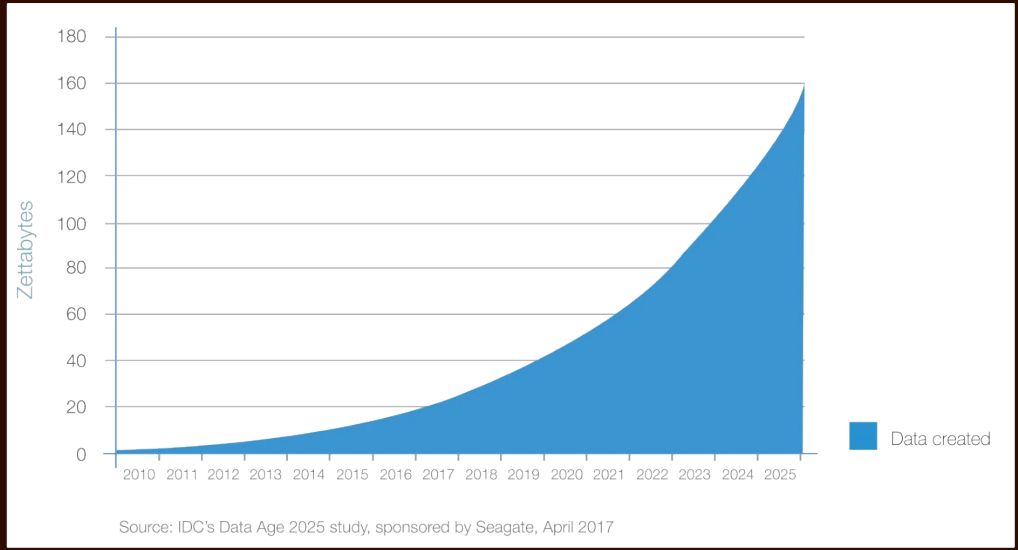


# CZYM JEST BIG DATA?

# ROCZNA WIELKOŚĆ GLOBALNEJ DATASFERY

Oszacowanie z roku 2017 wg IDC

(1ZB = 10^21 B)




**44ZB**

Ilość danych, które miały powstać w 2020 wg szacunków z 2017 r.

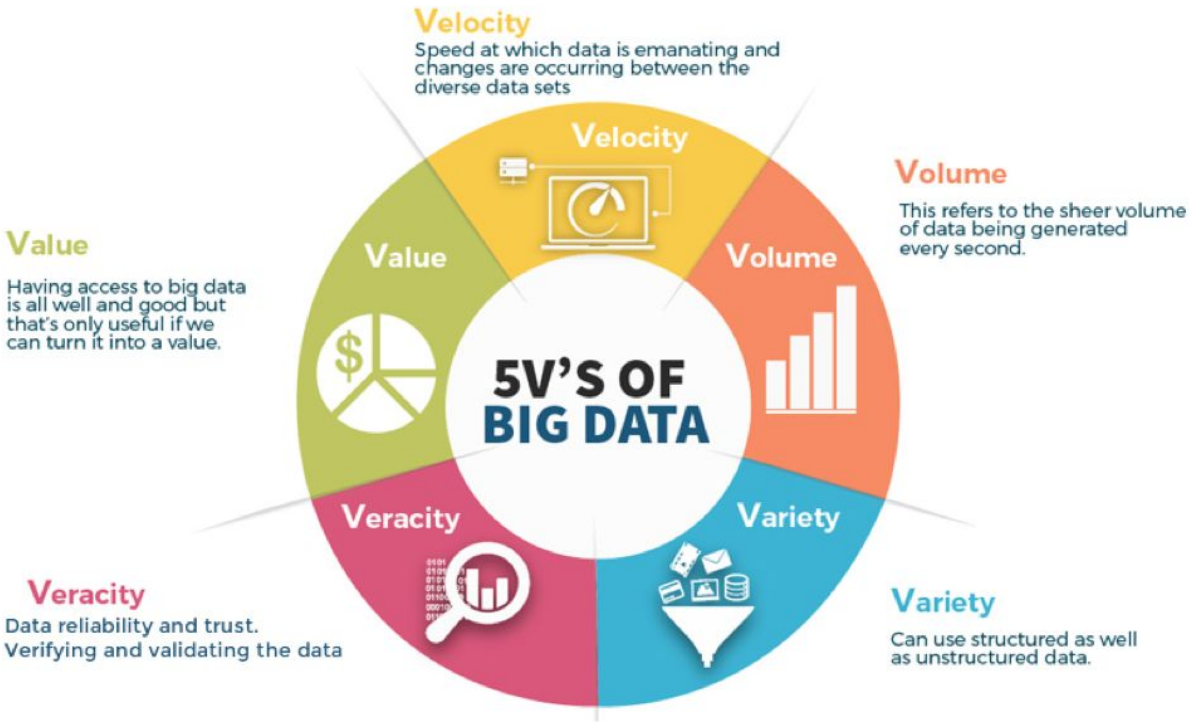
**64ZB**

Ilość danych, które powstały w 2020 wg szacunków z 2021 r.

**163ZB**

Ilość danych, które powstaną w 2025 wg szacunków z 2017 r.

# V's of BIG DATA



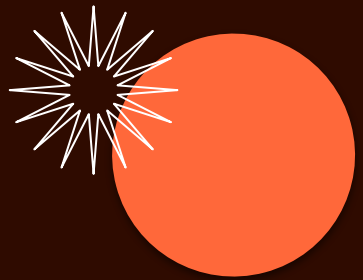
# KIEDY RAMKA DANYCH NIE MIEŚCI SIĘ W PAMIĘCI


# APACHE SPARK


- Napisany w Scali z pewnym wsparciem dla Pythona i R. Dobrze współpracuje z innymi JVM.
- Kompleksowy projekt z własnym ekosystemem.
- Dobrze integruje się z innymi projektami Apache.
- Dominujące i cieszące się zaufaniem narzędzie w świecie korporacyjnego Big Data.
- Rozszerzenie paradygmatu Map-Shuffle-Reduce.
- Wewnętrzny model Sparka zapewnia dobrą optymalizację w obliczenia równoległych.
- Skaluje się od pojedynczego węzła do klastrów z tysiącami węzłów.
- Posiada własny interfejs i model pamięci ramek danych (Spark DataFrames)
- Implementuje duży podzbiór języka SQL oraz zawiera optymalizator dla złożonych zapytań.
- Spark MLLib to spójny projekt uczenia maszynowego z obsługą typowych operacji.
- Mini-batchowa obsługa strumieniowego przesyłania danych o przyzwoitej wydajności.
- Nie zawiera natywnej obsługi tablic wielowymiarowych.
- Idealny dla typowych przypadków ETL + SQL i gdy potrzeba opakować wszystko w 1 projekt.

# DASK


- Napisany w Pythonie. Obsługuje tylko Pythona.
- Składnik większego naukowego ekosystemu Pythona, tj. NumPy, pandas i scikit-learn.
- Stworzony w 2014 roku.
- Wewnętrzny model Dask opiera na się na ogólnym harmonogramowaniu zadań.
- Model Dask jest niższego poziomu niż model Spark, brak optymalizacji wysokiego poziomu.
- Umożliwia budowanie bardziej skomplikowanych rozwiązań.
- Dask skaluje się od pojedynczego węzła do klastrów z tysiącami węzłów.
- Dask DataFrame wykorzystuje interfejs Pandas API i model pamięci.
- Nie implementuje ani SQL, ani optymalizatora zapytań.
- Współpracuje z istniejącymi bibliotekami ML oraz dostarcza własnej biblioteki dask-ml.
- W pełni obsługuje model NumPy dla skalowalnych tablic wielowymiarowych .
- Zapewnia interfejs futures w czasie rzeczywistym, który jest niższego poziomu niż przesyłanie strumieniowe Spark. Daje to więcej możliwości, ale wymaga więcej pracy.
- Wskazany dla rozwiązań wychodzących poza standardowy ETL+SQL i/lub zintegrowanych z naukowym ekosystemem Pythona.



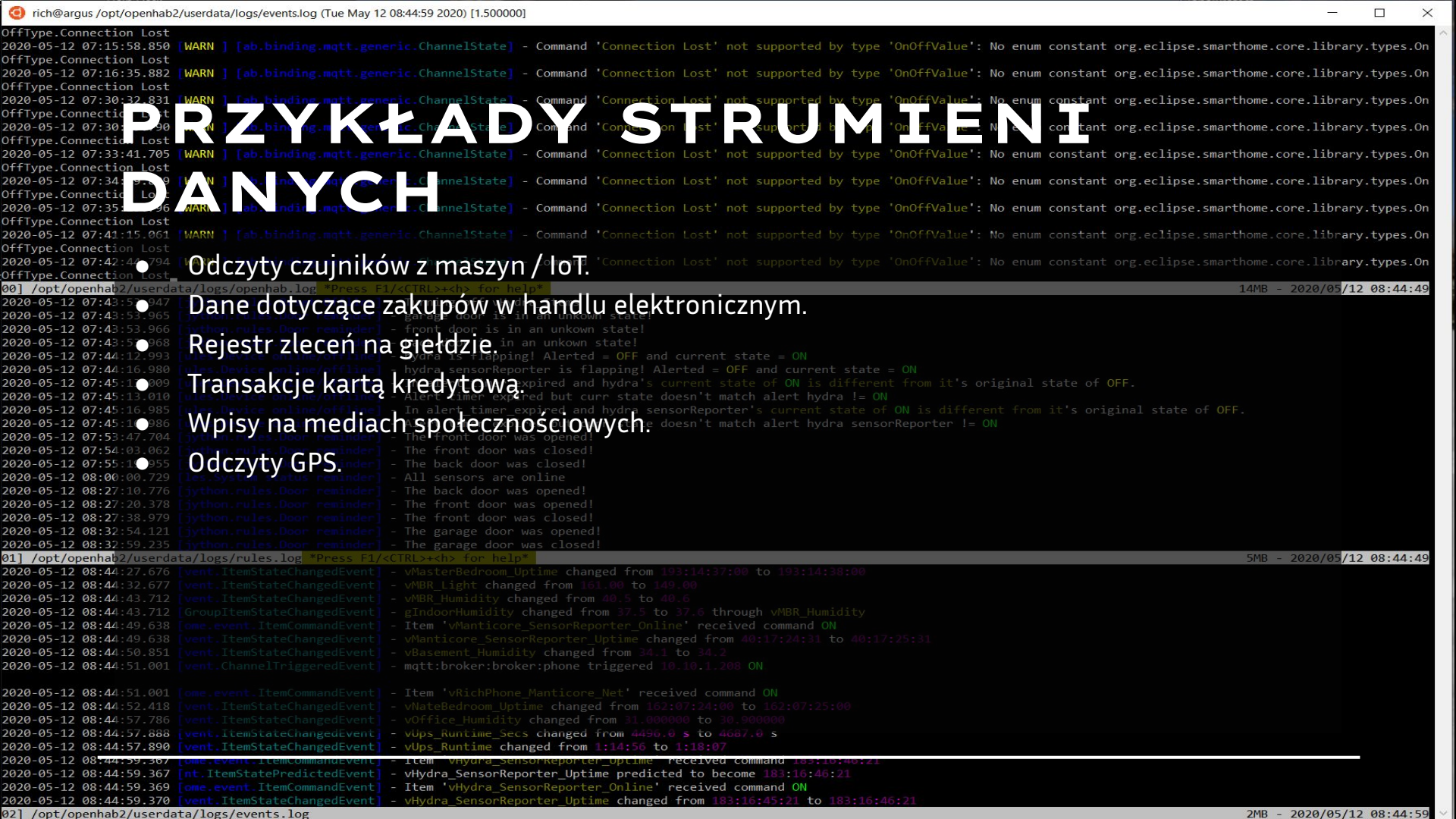
02

# ANALIZA DANYCH STRUMIENIOWYCH

# PRZETWARZANIE WSADOWE I STRUMIENIOWE

- **Przetwarzanie wsadowe -** przetwarzanie i analiza odbywa się na zbiorze danych, które były już przechowywane przez pewien czas. Dane są grupowane i przetwarzane w regularnych odstępach czasu.
    - Przykładem są systemy płacowe i rozliczeniowe, które muszą być przetwarzane co tydzień lub co miesiąc.

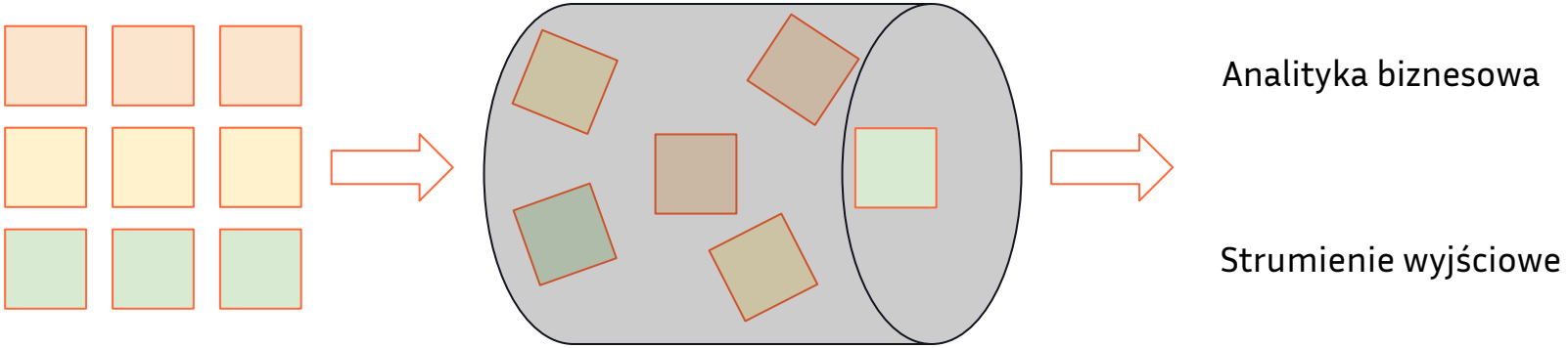
- **Przetwarzanie strumieniowe** - ciągły przepływ danych generowanych przez różne źródła. Korzystając z technologii przetwarzania strumieniowego, strumienie danych mogą być przetwarzane, przechowywane, analizowane i przetwarzane w miarę ich generowania w czasie rzeczywistym.


rich@argus /opt/openhab2/userdata/logs/events.log (Tue May 12 08:44:59 2020) [1.500000]
PRZYKŁADY STRUMIENI DANYCH
Odczyty czujników z maszyn / IoT.
Dane dotyczące zakupów w handlu elektronicznym.
Rejestr zleceń na giełdzie.
Transakcje kartą kredytową.
Wpisy na mediach społecznościowych.
Odczyty GPS.

# PRZYKŁADY ZASTOSOWAŃ

- SIEM (Security Information and Event Management):
  - analizowanie logów i dzienników zdarzeń w czasie rzeczywistym w celu monitorowania, metryk i wykrywania zagrożeń
- Zapasy w handlu detalicznym/magazynie:
  - zarządzanie zapasami we wszystkich kanałach i lokalizacjach oraz zapewnienie bezproblemowej obsługi na wszystkich urządzeniach
- Dopasowywanie udziału w przejazdach:
  - Łączenie danych dotyczących lokalizacji, użytkowników i cen w celu analizy predykcyjnej — dopasowywanie pasażerów do najlepszych kierowców pod względem odległości, miejsca docelowego, cen i czasu oczekiwania
- Uczenie maszynowe i sztuczna inteligencja:
  - Łącząc przeszłe i obecne dane dla jednego modelu (continual learning AKA incremental learning)

# SYSTEMY PRZETWARZANIA DANYCH STRUMIENIOWYCH

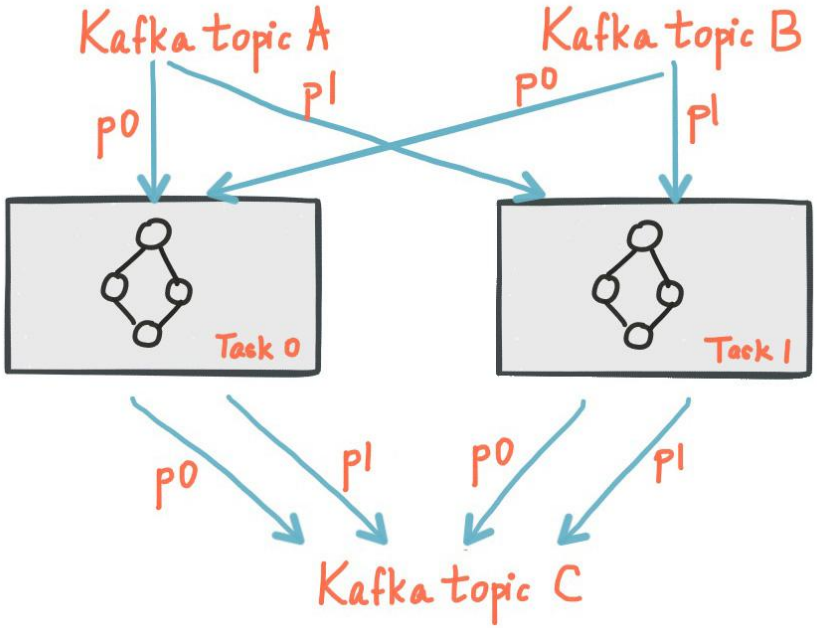


# NARZĘDZIA

# APACHE KAFKA


- Dane przetwarzane w wieloetapowych potokach (ang. pipeline), w których:
  - surowe dane wejściowe są pobierane z tematów Kafki,
  - następnie agregowane, wzbogacane lub w inny sposób przekształcane w nowe tematy (ang. topic) do dalszego wykorzystania lub dalszego przetwarzania.
- Przykłady:
  - potok przetwarzania polecający artykuły z wiadomościami może indeksować treść artykułu z kanałów RSS i publikować ją w temacie „artykuły”;
  - dalsze przetwarzanie może znormalizować lub zdeduplikować tę treść i opublikować oczyszczoną treść artykułu w nowym temacie;
  - końcowy etap przetwarzania może próbować polecić tę treść użytkownikom.

# APACHE KAFKA – PRZYKŁAD



# APACHE STORM

## Why use Apache Storm?

Apache Storm is a free and open source distributed realtime computation system. Apache Storm makes it easy to reliably process unbounded streams of data, doing for realtime processing what Hadoop did for batch processing. Apache Storm is simple, can be used with any programming language, and is a lot of fun to use!

Apache Storm has many use cases: realtime analytics, online machine learning, continuous computation, distributed RPC, ETL, and more. Apache Storm is fast: a benchmark clocked it at over **a million tuples processed per second per node**. It is scalable, fault-tolerant, guarantees your data will be processed, and is easy to set up and operate.

Apache Storm integrates with the queueing and database technologies you already use. An Apache Storm topology consumes streams of data and processes those streams in arbitrarily complex ways, repartitioning the streams between each stage of the computation however needed. Read more in the tutorial.

# APACHE FLINK

Flink

- What is Apache Flink?
  - Architecture
  - Applications
  - Operations
- What is Stateful Functions?
- What is Flink ML?
- What is the Flink Kubernetes Operator?
- What is Flink Table Store?
- Use Cases
- Powered By
- Downloads
- Getting Started
- Documentation

## What is Apache Flink? — Architecture

Apache Flink is a framework and distributed processing engine for stateful computations over *unbounded and bounded* data streams. Flink has been designed to run in *all common cluster environments*, perform computations at *in-memory speed* and at *any scale*.

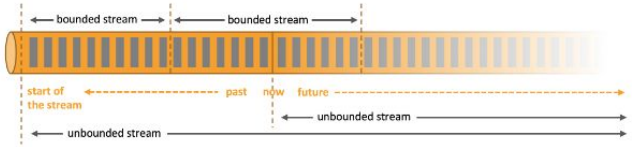
Here, we explain important aspects of Flink's architecture.

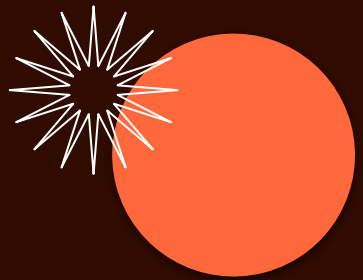
### Process Unbounded and Bounded Data

Any kind of data is produced as a stream of events. Credit card transactions, sensor measurements, machine logs, or user interactions on a website or mobile application, all of these data are generated as a stream.

Data can be processed as *unbounded* or *bounded* streams.

1. **Unbounded streams** have a start but no defined end. They do not terminate and provide data as it is generated. Unbounded streams must be continuously processed, i.e., events must be promptly handled after they have been ingested. It is not possible to wait for all input data to arrive because the input is unbounded and will not be complete at any point in time. Processing unbounded data often requires that events are ingested in a specific order, such as the order in which events occurred, to be able to reason about result completeness.
2. **Bounded streams** have a defined start and end. Bounded streams can be processed by ingesting all data before performing any computations. Ordered ingestion is not required to process bounded streams because a bounded data set can always be sorted. Processing of bounded streams is also known as batch processing.



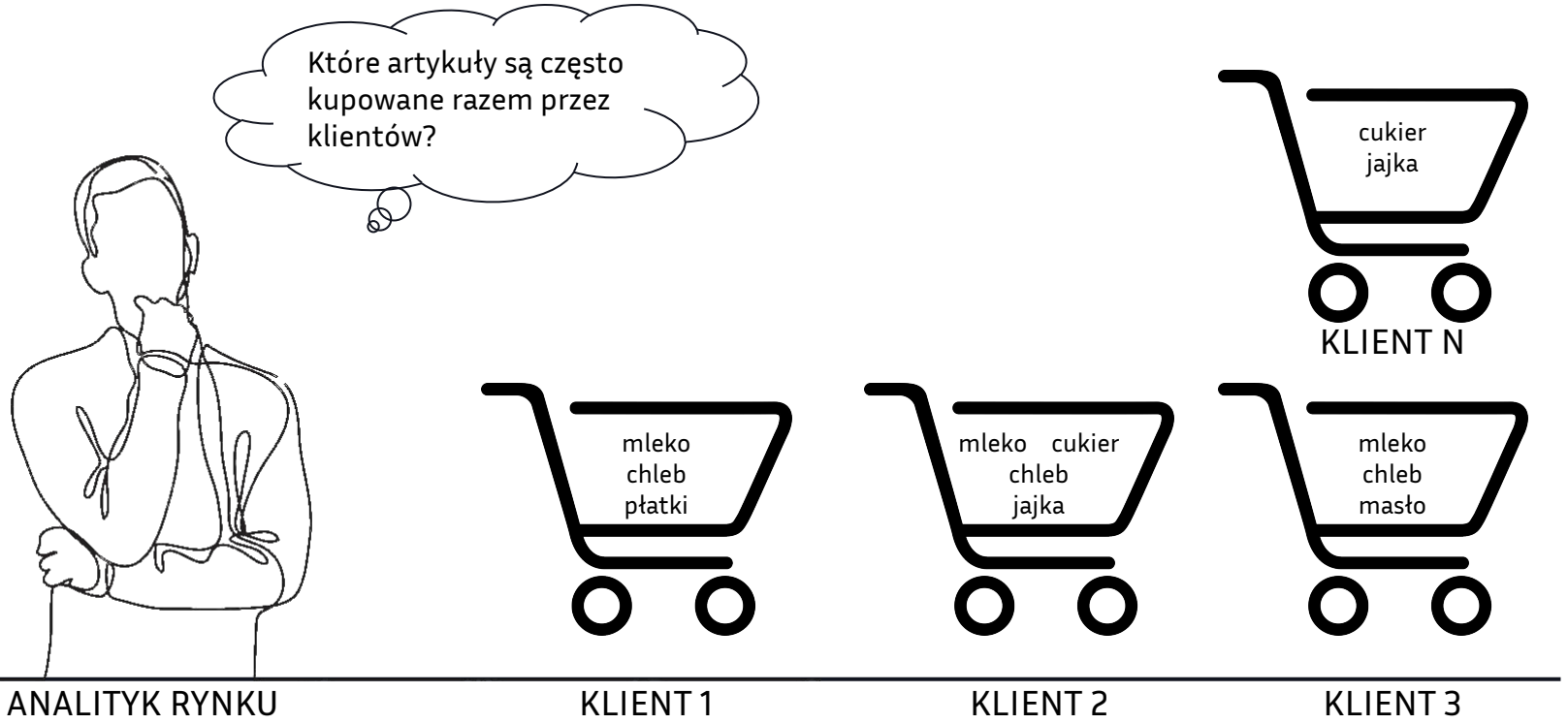


03

# ODKRYWANIE WZORCÓW SEKWENCJI

# PRZYPOMNIENIE: ODKRYWANIE ASOCJACJI

# ODKRYWANIE ASOCJACJI



# ODKRYWANIE TYPOWYCH OBJAWÓW CHOROBY

Comput Biol Med. 2021 Apr; 131: 104249.
Published online 2021 Feb 1. doi: 10.1016/j.compbiomed.2021.104249
PMCID: PMC7966840
PMID: 33561673

## Discovering symptom patterns of COVID-19 patients using association rule mining

Meera Tandan, PhD,[a,*] Yogesh Acharya, MD,[b] Suresh Pokharel, PhD,[c] and Mohan Timilsina, PhD[d]

Journal of Safety Research
Volume 76, February 2021, Pages 73-82

# Data mining approach to model bus crash severity in Australia

Seyed Alireza Samerei [a], Kayvan Aghabayk [a], Amin Moh

Show more

Share Cite

https://doi.org/10.1016/j.jsr.2020.12.004

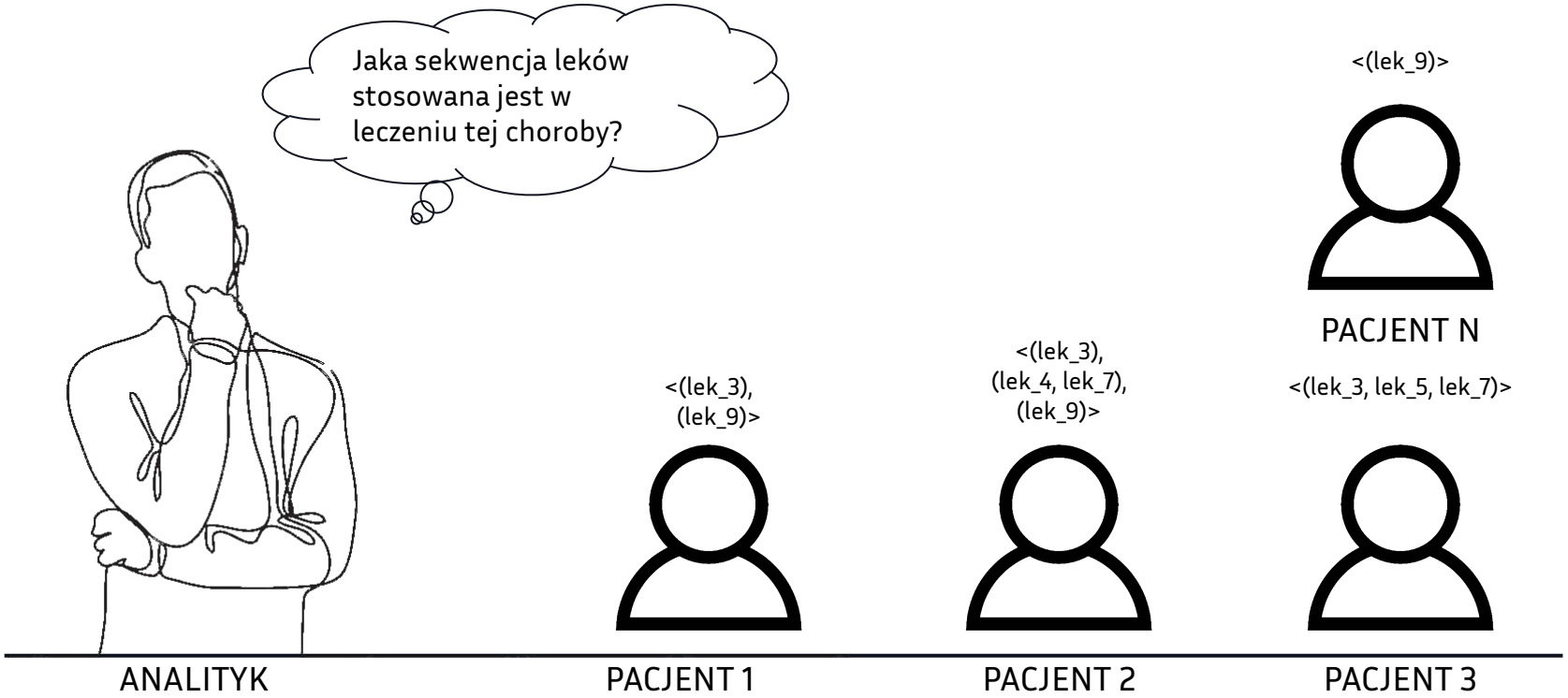
## Abstract

*Introduction:* Buses are different vehicles in terms of dimensions, maneuverability, and driver's vision. Although bus traveling is a safe mode to travel, the number of annual bus crashes cannot be neglected. Moreover, limited studies have been conducted on the bus involved in fatal crashes. Therefore, identification of the contributing factors in the bus involved fatal crashes can reduce the risk of fatality. *Method:* Data set of bus involved crashes in the State of Victoria, Australia was analyzed over the period of 2006–2019. Clustering of crash data was accomplished by dividing them into homogeneous categories, and by implementing association rules discovery on the clusters, the factors affecting fatality in bus involved crashes were extracted. *Results:* Clustering results show bus crashes with all vehicles except motor vehicles and weekend crashes have a high rate of fatality. According to the association rule discovery findings, the factors that increase the risk of bus crashes with non-motor vehicles are: old bus driver, collision with pedestrians at signalized intersections, and the presence of vulnerable road users. Likewise, factors that

https://www.sciencedirect.com/science/article/abs/pii/S002243752030150X

# ODKRYWANIE WZORCÓW SEKWENCJI

# ODKRYWANIE WZORCÓW SEKWENCJI



# ODKRYWANIE WZORCÓW SEKWENCJI

- Eksploracja wzorców sekwencyjnych ma szerokie zastosowanie
  - Sekwencje zakupów klientów
    - Najpierw kup laptopa, potem aparat cyfrowy, a następnie smartfon w ciągu 6 miesięcy
  - Zabiegi medyczne, klęski żywiołowe (np. trzęsienia ziemi), procesy naukowe i inżynieryjne, akcje i rynki, ...
  - Strumienie kliknięć w blogu, wzorce połączeń, ...
  - Inżynieria oprogramowania: sekwencje wykonywania programów, ...
  - Sekwencje biologiczne: DNA, białek, ...
- Wzorce sekwencyjne z przerwami i bez przerw

# PRZYKŁAD: ODKRYWANIE PROCESU

*“Złożoność i sztywność starszego (ang. legacy) oprogramowania w dużych organizacjach powoduje sytuacje, w których* ***pracownicy muszą wykonywać powtarzalne czynności*** *w celu przesyłania danych z jednej aplikacji do drugiej za pośrednictwem interfejsów użytkownika, np. przenoszenie danych z arkusza kalkulacyjnego do aplikacji webowej lub odwrotnie.* ***Odkrywanie i automatyzacja takich procedur*** *może pomóc wyeliminować żmudną pracę, skrócić czas cyklu i poprawić jakość danych. “*

- Eksploracja procesów - obszar badań skoncentrowany na zrozumieniu procesów.
- Opiera się na rzeczywistej realizacji procesu, a nie wyidealizowanych modelach.
- Ma na celu odkrywanie, sprawdzanie i ulepszanie rzeczywistych procesów biznesowych ze zdarzeń dostępnych w wielu systemach, np. w dziennikach zdarzeń.

# WZORCE SEKWENCJI

- **Sekwencja** - uporządkowana lista $S = \langle T_1 T_2 \dots T_n \rangle$, gdzie $T_i$ jest niepustym podzbiorem zbioru elementów $L = \{l_1, l_2, \dots, l_m\}$.
- $T_i = (x_1 x_2 \dots x_l)$ nazywamy **wyrazem sekwencji**.
- Dowolny element $x_j$ może wystąpić **tylko raz w danym wyrazie** sekwencji S, ale wielokrotnie w całej sekwencji.
- **Długość sekwencji -** liczba wyrazów w S.
- **Rozmiar sekwencji** - liczba elementów w S.
- **Przykład**: <(7) (3, 8) (9) (3, 5, 6)>

- Mówimy, że wyraz T sekwencji S wspiera zbiór $X \subseteq L$, jeżeli T zawiera każdy element ze zbioru X, $X \subseteq T$.
- **Zbiór częsty** - Zbiór X, którego wsparcie jest większe od zadanej minimalnej wartości progowej *min_sup*.

# ODKRYWANIE SEKWENCJI

**Odkrywanie sekwencji** - mając daną bazę danych sekwencji, znajdź pełny zestaw częstych podciągów, tj. spełniających próg *min_sup*.

Sekwencja: <a(abc)(ac)d(cf)>

| S_ID | Sekwencja |
|---|---|
| A | <a(abc)(ac)d(cf)> |
| B | <(ad)c(bc)(ae)> |
| C | <(ef)(ab)(df)cb> |
| D | <eg(af)cbc> |

Baza danych sekwencji

Podciąg sekwencji: <a(bc)dc> jest podciągiem sekwencji <**a**(a**bc**)(ac)**d**(**c**f)>

Przyjmując próg wsparcia min_sup=2, <(ab)c> jest wzorcem sekwencyjnym.

# SEKWENCJE WSPIERAJĄCE

Która z poniższych sekwencji nie zawiera się w żadnej sekwencji w bazie danych?

- <abcd>
- <aa(bc)>
- <f(ba)fb>
- <eg(af)cbc>

| S_ID | Sekwencja |
|---|---|
| A | <a(abc)(ac)d(cf)> |
| B | <(ad)c(bc)(ae)> |
| C | <(ef)(ab)(df)cb> |
| D | <eg(af)cbc> |

- Sekwencja S wspiera sekwencję Q, jeżeli Q zawiera się w S.
- Mówimy, że sekwencja S jest **maksymalna,** jeżeli nie zawiera się w żadnej innej sekwencji.
- **Wsparcie sekwencji S** - iloraz liczby sekwencji wspierających S do liczby wszystkich sekwencji.

$$support(S, D_s) = \frac{\{S_i \in D_s \ : S_i \text{ wspiera } S\}}{|D_s|}$$

Uwaga! Wsparcie obliczane jest względem liczby sekwencji, a nie liczby transakcji.

# ALGORYTMY ODKRYWANIA WZORCÓW SEKWENCJI

- Wymagania względem algorytmów:
  - wydajny, skalowalny, będący w stanie znaleźć pełny zbiór sekwencji, umożliwiający użytkownikom wprowadzenie różnych rodzajów ograniczeń.
- Sekwencje spełniają własność **antymonotoniczności**: Jeżeli sekwencja $s_1$ jest nieczęsta, żadna z sekwencji zawierających $s_1$ nie może być częsta.
- Reprezentacyjne algorytmy:
  - **GSP** (Generalized Sequential Patterns) - rozszerzenie idei algorytmu Apriori
  - Algorytm eksploracji pionowej: **SPADE** - rozszerzenie idei algorytmu ECLAT
  - Metoda przyrostu wzorca: **PrefixSpan** - rozszerzenie idei algorytmu FP-Growth

# ALGORYTM ODKRYWANIA WZORCÓW SEKWENCJI

Wejście: baza danych transakcji D, minimalne wsparcie minsup
Wyjście: zbiór wzorców sekwencji

1. **Sortowanie**
   a. Przekształcenie oryginalnej bazy do bazy danych sekwencji.
      i. np. posortwanie bazy danych recept wg *pacjent_id*, *recepta_timestamp*.
2. **Znajdowanie zbiorów częstych** w bazie danych sekwencji.
3. **Transformacja**
   a. Każdy wyraz T należący do sekwencji zamień w listę zbiorów częstych zawierających się w tym wyrazie.
4. **Sekwencjonowanie**
   a. Znajdź w bazie danych sekwencji wszystkie sekwencje częste (wzorce sekwencji), których wsparcie jest większe lub równe *minsup*.
5. **Maksymalizacja** (opcjonalna)
   a. Znajdź maksymalne wzorce sekwencji

# GSP: GENERALIZED SEQUENTIAL PATTERN

1. Niech k=1
2. Powtarzaj
   a. Stwórz zbiór kandydatów $C_k$ składający się z k-elementowych sekwencji częstych
   b. Oblicz wsparcie dla każdego zbioru w $C_k$
   c. Filtrowanie zbiorów kandydujących: $L_k$ - zbiór tych zbiorów c z $C_k$, dla których $s(c) \geqslant minsup$
   d. k := k + 1
   e. Jeżeli $L_{k-1} = \emptyset$, to przerwij pętlę.
3. Zwróć $\bigcup_k L_k$

# GSP: GENERALIZED SEQUENTIAL PATTERN

Algorytm GSP przypomina algorytm Apriori.
Główna różnica polega na sposobie generowania zbiorów kandydujących.

Załóżmy, że:

- A → B i A → C są dwoma częstymi 2-sekwencjami opartymi na zbiorach częstych (A,B) i (A, C).
- Zbiorem kandydującym w zwykłym Apriori byłby zbiór (A, B, C) jako zestaw 3-elementowy.
- W GSP otrzymujemy następujące 3-sekwencje: A → B → C, A → C → B i A → BC

# PrefixSpan

ang. Prefix-projected Sequential pattern mining

**Example 1 (Running example)** Let our running database be *sequence database* $S$ given in Table 1 and *min_support* = 2. The set of *items* in the database is $\{a, b, c, d, e, f, g\}$.

| Sequence_id | Sequence |
|---|---|
| 10 | $\langle a(abc)(ac)d(cf)\rangle$ |
| 20 | $\langle (ad)c(bc)(ae)\rangle$ |
| 30 | $\langle (ef)(ab)(df)cb\rangle$ |
| 40 | $\langle eg(af)cbc\rangle$ |

**Table 1.** A sequence database

For example, $\langle a\rangle$, $\langle aa\rangle$, $\langle a(ab)\rangle$ and $\langle a(abc)\rangle$ are *prefixes* of sequence $\langle a(abc)(ac)d(cf)\rangle$, but neither $\langle ab\rangle$ nor $\langle a(bc)\rangle$ is considered as a prefix. $\langle (abc)(ac)d(cf)\rangle$ is the *postfix* of the same sequence w.r.t. prefix $\langle a\rangle$, $\langle (\_bc)(ac)d(cf)\rangle$ is the *postfix* w.r.t. prefix $\langle aa\rangle$, and $\langle (\_c)(ac)d(cf)\rangle$ is the *postfix* w.r.t. prefix $\langle ab\rangle$.

Han, J., Pei, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., & Hsu, M. (2001, April). Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In proceedings of the 17th international conference on data engineering (pp. 215-224). IEEE.

# PrefixSpan

| Prefix | Projected (postfix) database | Sequential patterns |
|---|---|---|
| $\langle a \rangle$ | $\langle (abc)(ac)d(cf) \rangle$, $\langle (\_d)c(bc)(ae) \rangle$, $\langle (\_b)(df)cb \rangle$, $\langle (\_f)cbc \rangle$ | $\langle a \rangle$, $\langle aa \rangle$, $\langle ab \rangle$, $\langle a(bc) \rangle$, $\langle a(bc)a \rangle$, $\langle aba \rangle$, $\langle abc \rangle$, $\langle (ab) \rangle$, $\langle (ab)c \rangle$, $\langle (ab)d \rangle$, $\langle (ab)f \rangle$, $\langle (ab)dc \rangle$, $\langle ac \rangle$, $\langle aca \rangle$, $\langle acb \rangle$, $\langle acc \rangle$, $\langle ad \rangle$, $\langle adc \rangle$, $\langle af \rangle$ |
| $\langle b \rangle$ | $\langle (\_c)(ac)d(cf) \rangle$, $\langle (\_c)(ae) \rangle$, $\langle (df)cb \rangle$, $\langle c \rangle$ | $\langle b \rangle$, $\langle ba \rangle$, $\langle bc \rangle$, $\langle (bc) \rangle$, $\langle (bc)a \rangle$, $\langle bd \rangle$, $\langle bdc \rangle$, $\langle bf \rangle$ |
| $\langle c \rangle$ | $\langle (ac)d(cf) \rangle$, $\langle (bc)(ae) \rangle$, $\langle b \rangle$, $\langle bc \rangle$ | $\langle c \rangle$, $\langle ca \rangle$, $\langle cb \rangle$, $\langle cc \rangle$ |
| $\langle d \rangle$ | $\langle (cf) \rangle$, $\langle c(bc)(ae) \rangle$, $\langle (\_f)cb \rangle$ | $\langle d \rangle$, $\langle db \rangle$, $\langle dc \rangle$, $\langle dcb \rangle$ |
| $\langle e \rangle$ | $\langle (\_f)(ab)(df)cb \rangle$, $\langle (af)cbc \rangle$ | $\langle e \rangle$, $\langle ea \rangle$, $\langle eab \rangle$, $\langle eac \rangle$, $\langle eacb \rangle$, $\langle eb \rangle$, $\langle ebc \rangle$, $\langle ec \rangle$, $\langle ecb \rangle$, $\langle ef \rangle$, $\langle efb \rangle$, $\langle efc \rangle$, $\langle efcb \rangle$. |
| $\langle f \rangle$ | $\langle (ab)(df)cb \rangle$, $\langle cbc \rangle$ | $\langle f \rangle$, $\langle fb \rangle$, $\langle fbc \rangle$, $\langle fc \rangle$, $\langle fcb \rangle$ |

**Table 2.** Projected databases and sequential patterns

Han, J., Pei, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., & Hsu, M. (2001, April). Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In proceedings of the 17th international conference on data engineering (pp. 215-224). IEEE.

04

# ODKRYWANIE WZORCÓW W DANYCH STRUMIENIOWYCH

# Automatic Sequential Pattern Mining in Data Streams

Koki Kawabata*
ISIR, Osaka University
koki88@sanken.osaka-u.ac.jp

Yasuko Matsubara*
ISIR, Osaka University
yasuko@sanken.osaka-u.ac.jp

Yasushi Sakurai*
ISIR, Osaka University
yasushi@sanken.osaka-u.ac.jp

## ABSTRACT

Given a large volume of multi-dimensional data streams, such as that produced by IoT applications, finance and online web-click logs, how can we discover typical patterns and compress them into compact models? In addition, how can we incrementally distinguish multiple patterns while considering the information obtained from a pattern found in a streaming setting? In this paper, we propose a streaming algorithm, namely StreamScope, that is designed to find intuitive patterns efficiently from event streams evolving over time. Our proposed [illegible] the act. This sudden explosion of intelligent connected devices imposes new requirements on data stream mining, which include (a) real-time modeling, and (b) automatic mining. The ideal method should capture the dynamics of time-series data, efficiently and automatically over IoT data streams. It should also enable us to perform model estimation and advanced analytics, such as anomaly detection, forecasting, and pattern recognition, without any parameter tuning steps.

For example, consider a large collection of observations contin- [illegible] generated by sensors in an [illegible] factory. The most

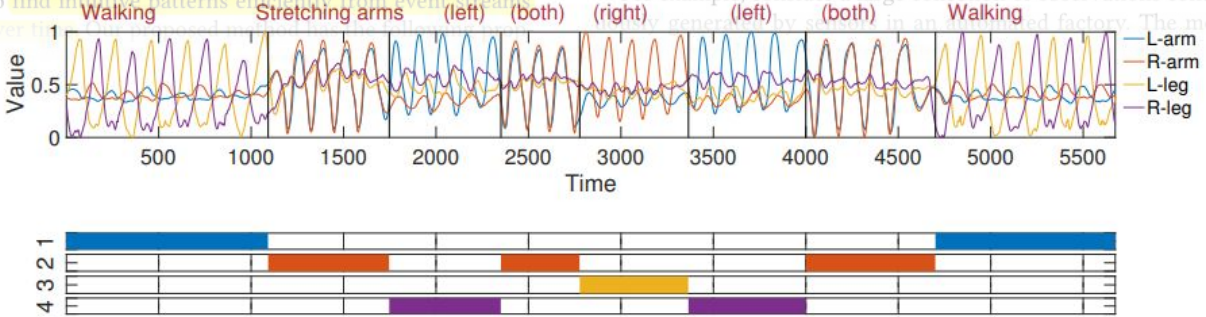



**Figure 1: Online pattern discovery with StreamScope: the original motion sensor streams (top), and the output of our method (bottom). At every time tick, the method incrementally identifies intuitive motions (such as walking and stretching arms) and classifies them into groups, namely regimes. If necessary, it automatically generates a new regime with no prior knowledge.**

Kawabata, K., Matsubara, Y., & Sakurai, Y. (2019, November). Automatic sequential pattern mining in data streams. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management (pp. 1733-1742).

# DANE STRUMIENIOWE

- **Dane strumieniowe** - ogromne ilości ciągłych danych, prawdopodobnie nieskończone.
  - Obraz świata szybko się zmienia i wymaga reakcji w czasie rzeczywistym.
  - Dla porównania - DBMS to skończone, trwałe zbiory danych odpowiadające pewnej “wersji” świata.
- Strumienie danych dobrze oddaje nasze dzisiejsze potrzeby w zakresie przetwarzania danych
- Wyzwania:
  - Losowy dostęp jest drogi:
    - Algorytm może mieć tylko jedno spojrzenie na dane (tzw. pojedyncze skanowanie)
  - W jaki sposób przechowywać dane?
    - Przechowuj tylko podsumowanie danych widzianych do tej pory (historycznych)
    - Pełnej analizie podlegają tylko “świeże” dane mieszczące się w predefiniowanym oknie.
- Problem dotyczy wszystkich metod eksploracji danych, nie tylko odkrywania sekwencji.

# LITERATURA

[1] Slimani, T., & Lazzez, A. (2013). Sequential mining: patterns and algorithms analysis. arXiv preprint arXiv:1311.0350. PDF
[2] Gama, J. (2010). Knowledge discovery from data streams. CRC Press. PDF
[3] Gaber, M. M., Zaslavsky, A., & Krishnaswamy, S. (2005). Mining data streams: a review. ACM Sigmod Record, 34(2), 18-26. PDF

THANKS!

# DZIĘKUJĘ ZA UWAGĘ