



UNIVERSITÄT
LEIPZIG

Big-Data-Praktikum SS 2019

Universität Leipzig, Institut für Informatik
Abteilung Datenbanken
Prof. Dr. E. Rahm



Database Group Leipzig

Department of Computer Science



Organisation

- ▶ **Ziel:** Entwurf und Realisierung einer Anwendung / Algorithmus unter Verwendung eines Big-Data-Frameworks (Apache Flink, Apache Spark, Hadoop)
- ▶ **Ablauf:**
 - ▶ Selbstständige Bearbeitung der Aufgabe (üblicherweise) in **2er-Gruppen**
 - ▶ **Testat 1:** System kennenlernen, Datenimport, Lösungsskizze
 - ▶ Abgabe: **Entwurfsdokument** (ca. 4 Seiten)
 - ▶ Deadline: 31.05.2019
 - ▶ **Testat 2:** Implementierung und Evaluierung
 - ▶ Abgabe: dokumentiertes, ausführbares **Programm**
 - ▶ Deadline: 31.07.2019
 - ▶ **Testat 3: Präsentation** (ca. 10+5 min pro Gruppe)
 - ▶ Termin: tba
 - ▶ **Anwesenheitspflicht aller Praktikumssteilnehmer zu allen Testaten!**

Technische Details

- ▶ Quellcode: Versionierung via [GitLab \(https://git.informatik.uni-leipzig.de\)](https://git.informatik.uni-leipzig.de)
- ▶ Java: [Apache Maven](#) für Projekt-Management
- ▶ Test-getriebene Entwicklung erwünscht
- ▶ Dokumentation des Quellcodes zwingend erforderlich
- ▶ Stabile Versionen der Frameworks verwenden
 - ▶ [Apache Flink 1.7.2](#)
- ▶ Lokal lauffähige Versionen können auf Computer-Cluster ausgeführt werden
 - ▶ Individuelle Absprache mit dem jeweiligen Betreuer
 - ▶ Notwendig zur Evaluierung von Skalierbarkeit / Speedup

Themenvergabe

- ▶ Angabe der Top-4-Wunsch-Themen
- ▶ Optional: Wunsch-Partner
 - ▶ Top-4-Themen sollten übereinstimmen
- ▶ Vergabe der Plätze durch Zuordnungsalgorithmus, z. B.
 - ▶ Stable-Marriage-Algorithmus
 - ▶ Kuhn-Munkres-Algorithmus

Eigene Angaben			Angaben Wunsch-Partner (Optional)		Nr. Wunsch-Themen (Top-4)
Matrikel	Nachname	E-Mail-Adresse	Matrikel	Nachname	
		@studserv.uni-leipzig.de			

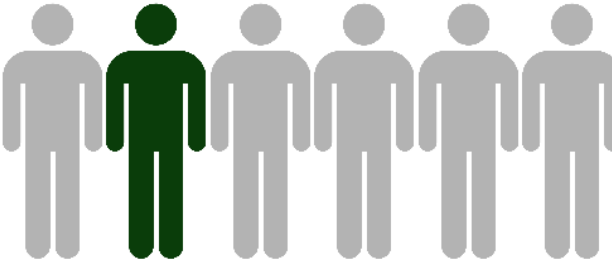
(Incremental) PPRL on Flink

Martin Franke

(Incremental) PPRL on Flink

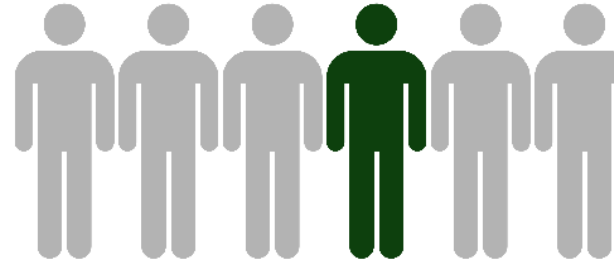
- ▶ Find records in different databases that refer to the same real-world entity
 - ▶ Persons, e.g., patients, customers
- ▶ No disclosure of sensitive personal information
 - ▶ State of health, diseases, address

Database Party A



Name: **Klaus Schmidt**
DOB: **07/03/1969**
Address: Ritterstraße 9
04109 Leipzig

Database Party B



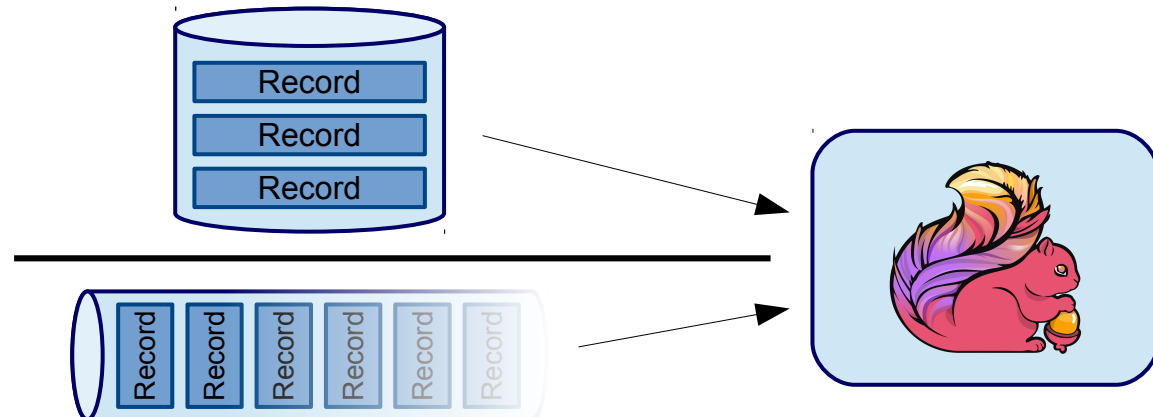
Name: **Claus Schmid**
DOB: **03.07.1969**
Address: Ritterstraße 9-13
04109 Leipzig

(Incremental) PPRL on Flink

- ▶ Potentially large databases containing millions of records
 - Scalability issues due to quadratic complexity of linkage
- ▶ Parallel and distributed PPRL using Apache Flink to speedup linkage
- ▶ **Tasks:**
 - ▶ Combine and refactor two existing Flink implementations for PPRL with metric space filtering and LSH-based blocking (Batch API)
 - ▶ Investigation of Flinks Streaming API for **incremental linkage** where records are continuously added

- ▶ **Technologies:**

- ▶ Java
- ▶ Apache Flink

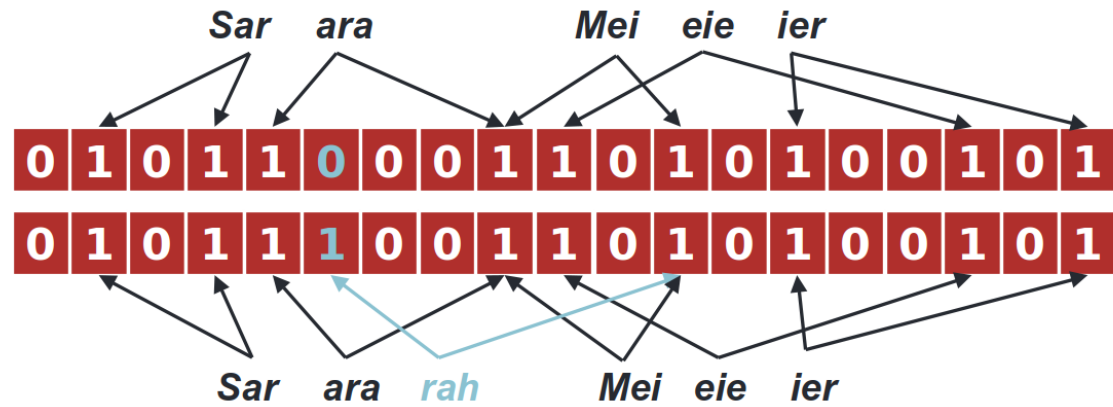


Cryptanalysis for Bloom Filter based Privacy-Preserving Record Linkage

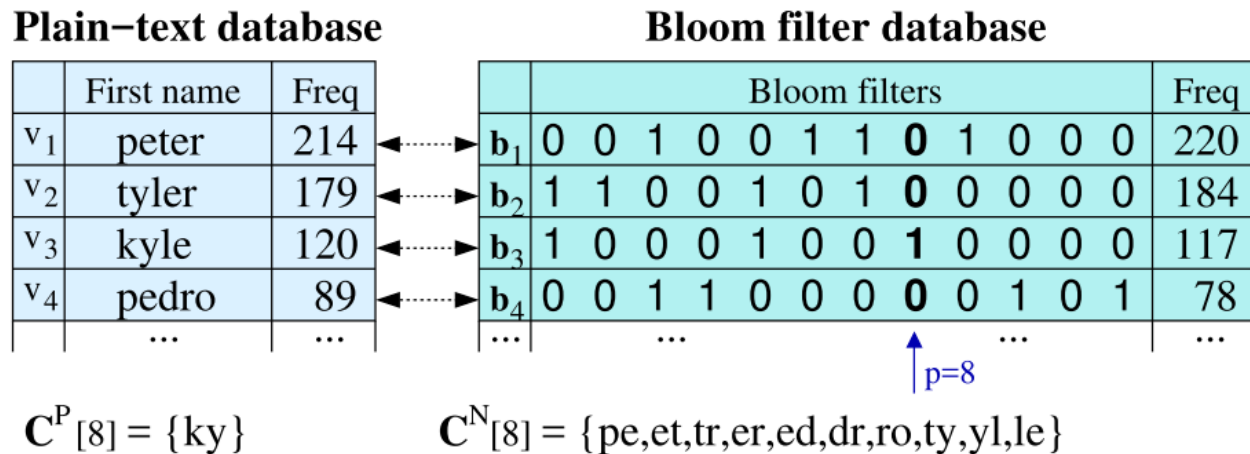
Florens Rohde

Cryptanalysis for Bloom-Filter-based PPRL

- ▶ Many PPRL protocols are based on Bloom Filter encodings of sensitive data because they allow probabilistic & efficient matching of huge datasets
- ▶ However Bloom Filter are known to be vulnerable to cryptanalysis



- ▶ Christen, Peter, et al. "Precise and Fast Cryptanalysis for Bloom Filter Based Privacy-Preserving Record Linkage." IEEE Transactions on Knowledge and Data Engineering (2018) <https://doi.org/10.1109/TKDE.2018.2874004>



Cryptanalysis for Bloom Filter based PPRL

▶ **Interesting questions:**

- ▶ How many attributes/records can be re-identified?
- ▶ How many records are needed for a (partially) successful attack?
- ▶ How do different encoding settings affect the runtime/quality of the attack?

▶ **Technology:**

- ▶ Apache Flink

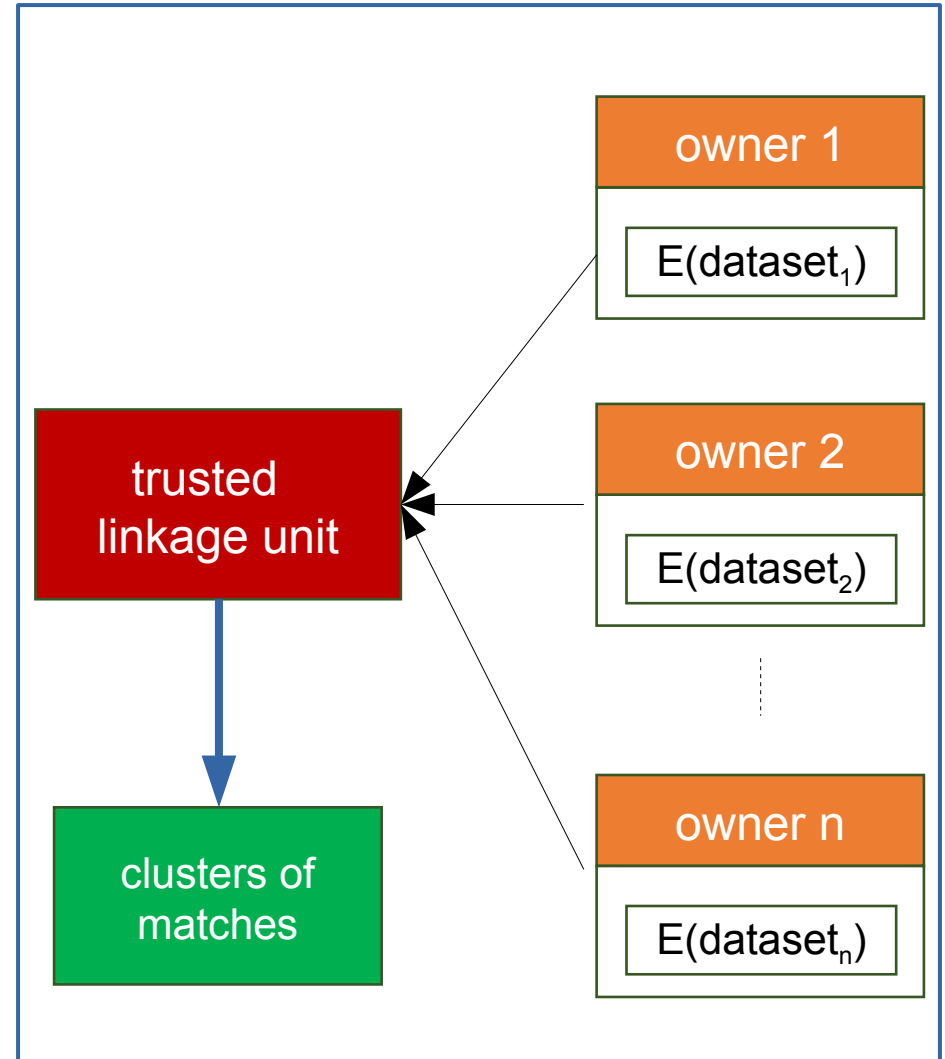
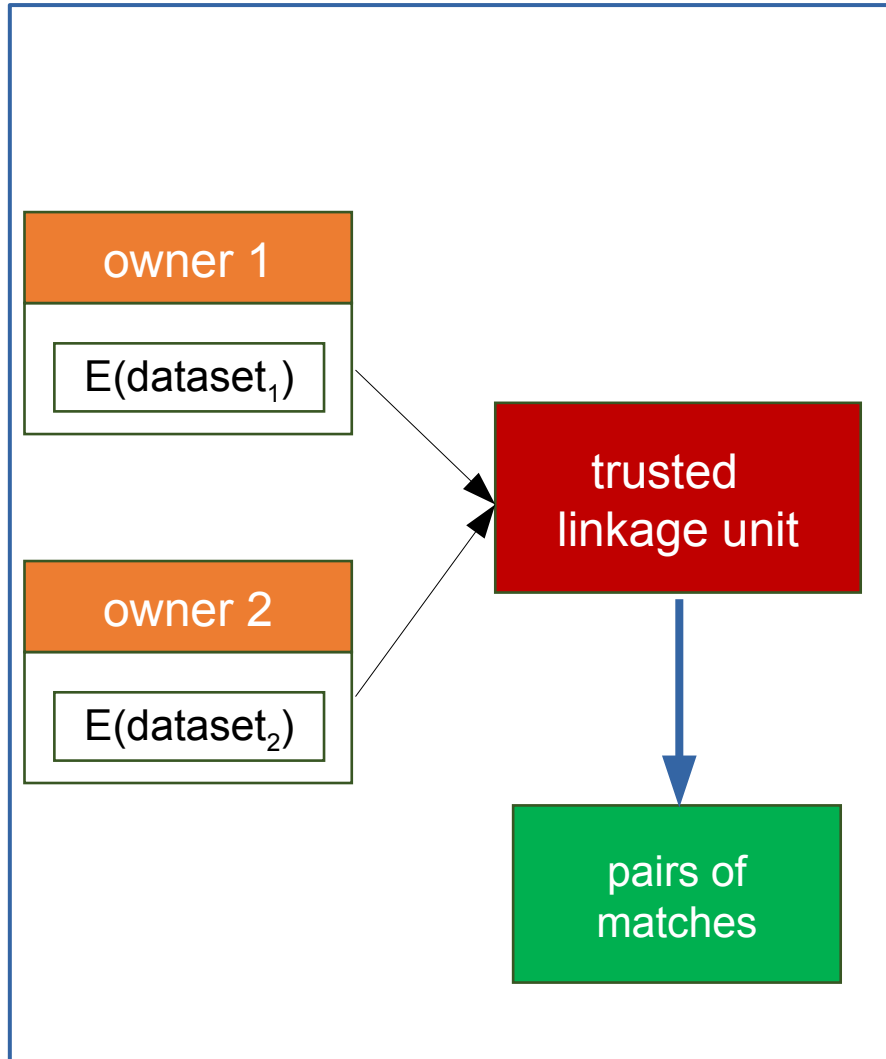
▶ **Datasets:**

- ▶ North Carolina Voter Register, Synthetic datasets

Parallel Multi-Party PPRL

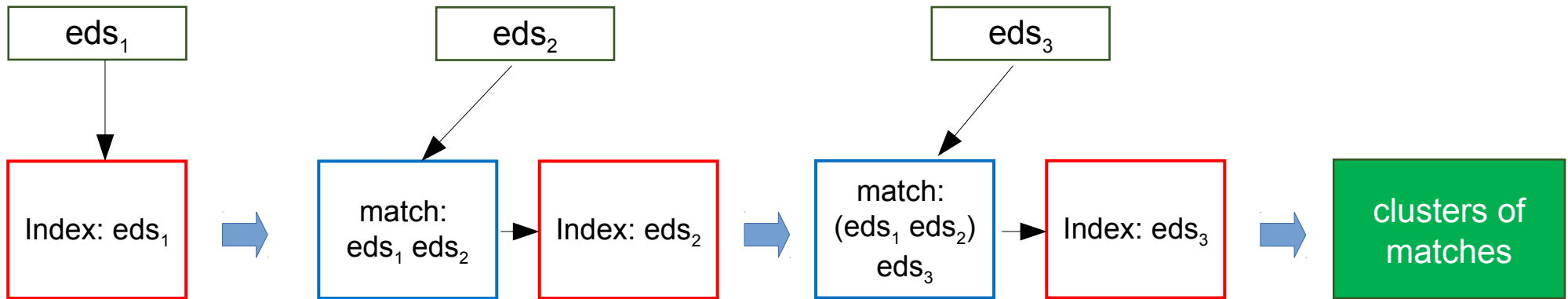
Ziad Sehili

PPRL vs. MP-PPRL

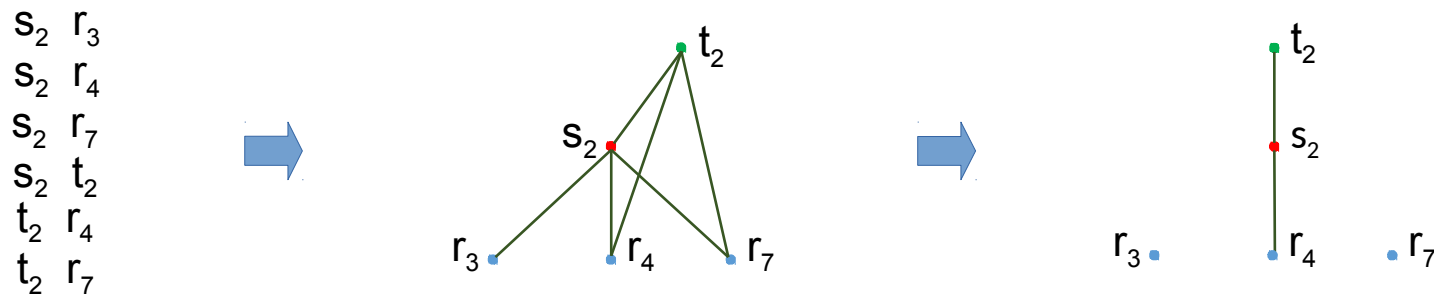


Steps to parallelize in MP-PPRL

► Matching



► Post-Processing:...

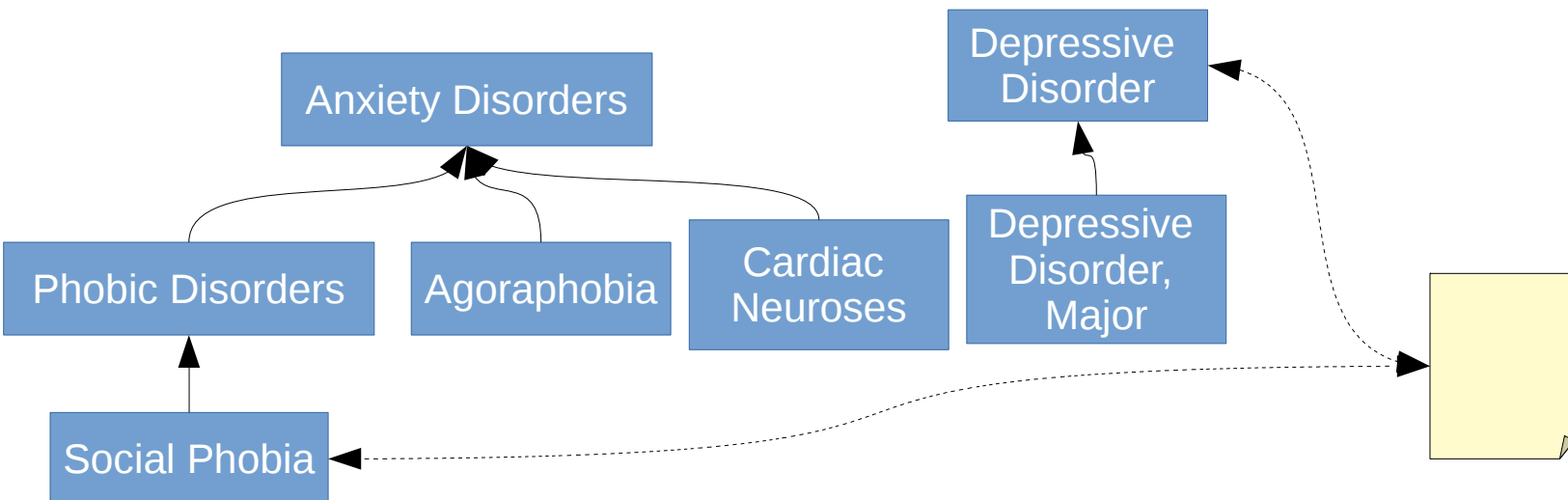


Rollup Querying of ontology graphs

Victor Christen

Rollup Querying of ontology graphs

- ▶ Ontologies describe real-world entities in a standardized way:
- ▶ Entities are linked to the most specific concept
- ▶ Users are interested in an overview or are not familiar with the data
- ▶ Querying on a high level according to the hierarchy of the ontology
- ▶ Result should contain entities linked with specialized concepts as well
 - ▶ Do Anxiety disorder and depressive disorder co-occur?



Rollup Querying of ontology graphs

- ▶ **Tasks:**

- ▶ Conceptualize an match operator considering the hierarchy of schema graphs
- ▶ Implement and evaluate the operator

- ▶ **Technology:**

- ▶ Gradoop, Flink

- ▶ **Data:**

- ▶ NCBIT corpus: Annotated biomedical abstracts with disease concepts

- ▶ **Previous Knowledge:**

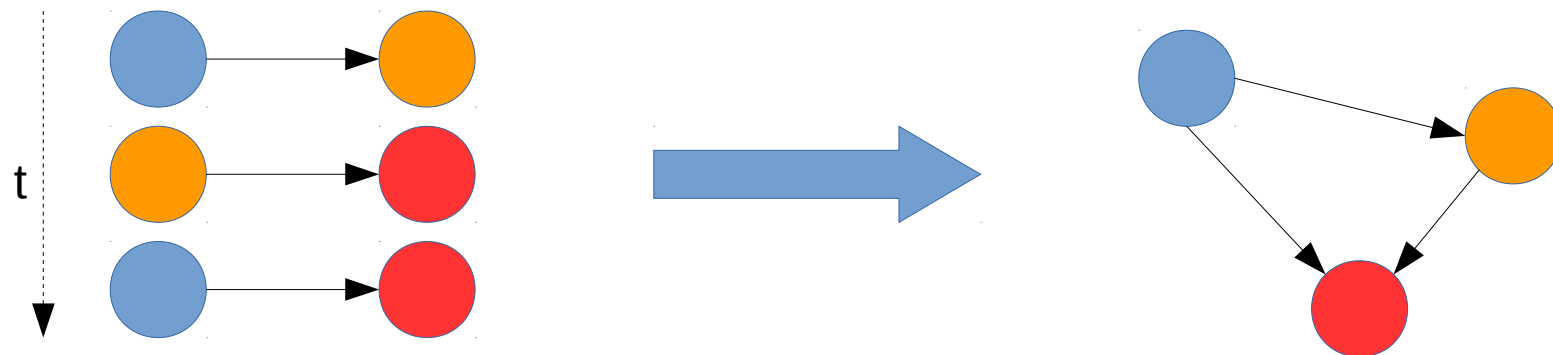
- ▶ Java

Processing and Visualization of Graph Streams

Christopher Rost

Proc. and Visualization of Graph Streams

- ▶ Real-world graphs evolve over time:
 - ▶ friendship networks on Facebook, community interactions at Stackoverflow, video-likes and channel-abos on YouTube, citation networks
- ▶ Edge stream as one typical representation
- ▶ Visualization helps observing the changing graph
- ▶ How do **communities** change over time?
- ▶ Are there **interesting patterns** within the graph at a given time?



Proc. and Visualization of Graph Streams

► Tasks:

- Create/choose a graph stream data source (real or synthetic)
- Build a **stream application** including windowing and grouping/aggregation
- **Visualize** the resulting graph snippet in a web application

► Technologies:

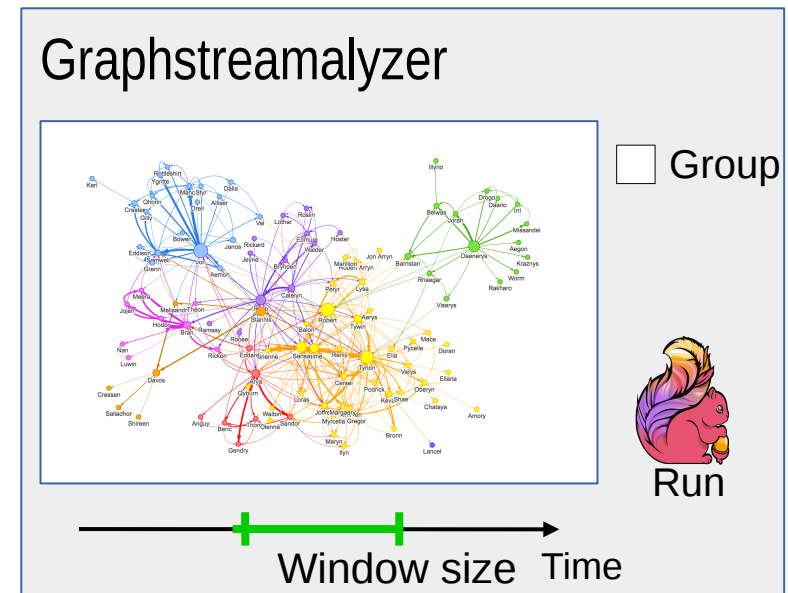
- **Streaming API** of Apache Flink

► Data:

- E.g. *Stanford Large Network Dataset Collection*

► Previous knowledge:

- Solid programming skills in Java/Scala



[Img] A. Beveridge and J. Shan, „Network of Thrones“ Math Horizons Magazine , Vol. 23, No. 4 (2016), pp. 18-22.

Distributed Graph Sampling using Forest Fire

Kevin Gomez

Forest Fire Sampling

▶ Graph Sampling?

- ▶ Graphs are potentially large (millions of nodes, billions of edges)
- ▶ Limitation from analyst regarding the size of the graphs
 - Create graph “sample” with equivalent content/structure

▶ Distributed?

- ▶ Size of a graph can exceed capacity of a single system
 - Distributed implementation on modern systems (Map-Reduce, Flink, Spark etc.) to allow processing and reduce runtimes

▶ Forest Fire (FF)?

- ▶ Algorithm known from Graph Evolution research
- ▶ Burning tree has probability to ignite adjacent trees

Forest Fire Sampling

- ▶ Question: Does FF create “good” samples?
- ▶ **Tasks:**
 1. Implementation of FF algorithm on Apache Flink (Gelly)
 2. Integration as operator in Gradoop
 3. Evaluation of operators on cluster with 16 worker nodes (speedup/scalability)
- ▶ **Technologies:**
 - ▶ Apache Flink, Gelly, Pregel, GRADOOP
- ▶ **Previous knowledge:**
 - ▶ Apache Flink, Gelly, Pregel, Maven, Git, Java

Representation Learning for Gradoop

Daniel Obraczka

Representation Learning for Gradoop

▶ **Tasks:**

- ▶ Creation of non implemented graph statistics
- ▶ Implementation of selection process for graph statistics

▶ **Technologies:**

- ▶ Java, Apache Flink, Gradoop

▶ **Data:**

- ▶ Stanford Large Network Dataset Collection ([SNAP](#)),
- ▶ Example graphs

▶ **Previous knowledge:**

- ▶ Java

Representation Learning for Gradoop

- ▶ Graphs are everywhere (social networks, recommender systems etc.)
- ▶ Machine Learning models can often not directly process graphs and need graph statistics as input (e.g. node degrees)
- ▶ Representation Learning produce embedded nodes in low dimensional vector space

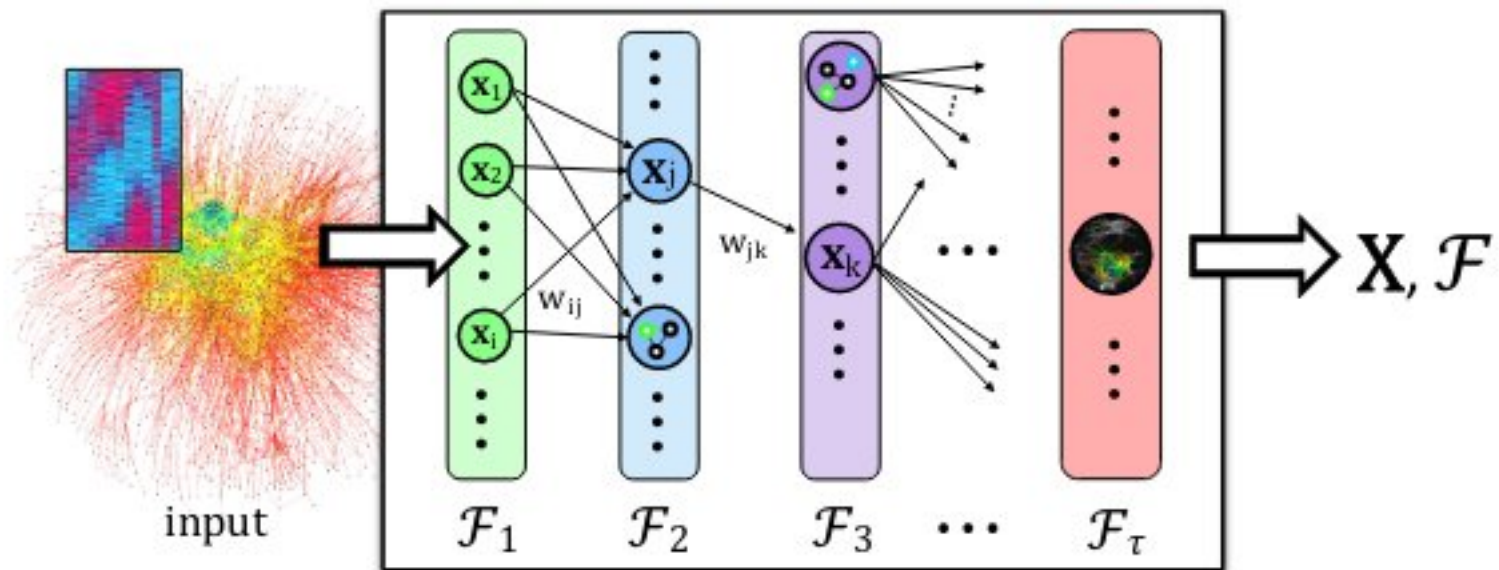


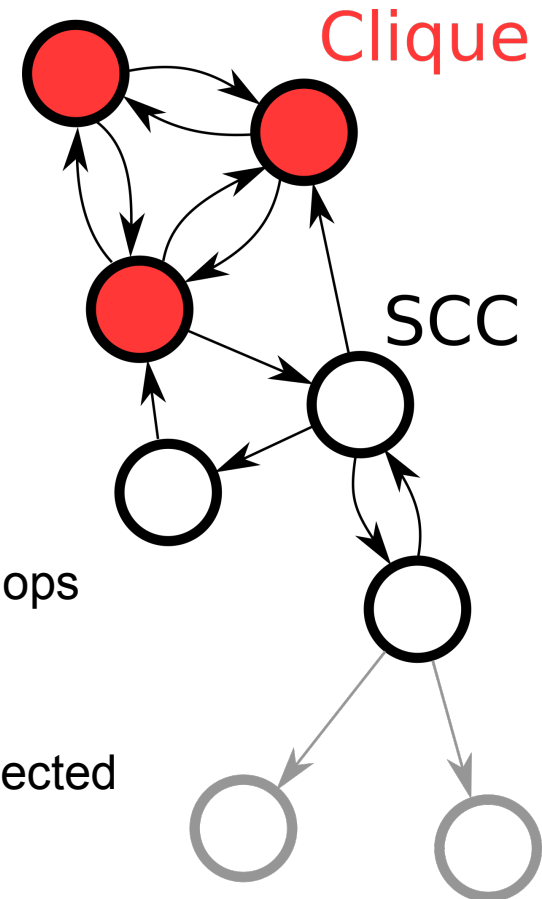
Bild: Deep Inductive Graph Representation Learning”, R.Rossi et. al. (<http://ryanrossi.com/pubs/rossi-et-al-TKDE18.pdf>)

Determination of connectivity in graphs

Matthias Täschner

Determination of connectivity in graphs

- ▶ Discover communities, their number and size
- ▶ Important for analysis of social networks or graph clustering
- ▶ Find (highly) linked sub-graphs, e.g.,
 - ▶ Strongly Connected Component (SCC)
 - ▶ Maximal cyclic sub-graph G_s for directed graph
 - ▶ For every pair of nodes (u,v) in G_s exists a directed path $P_{(v \rightarrow u)}(n)$ for an arbitrary number of hops
 - ▶ Clique
 - ▶ Maximal sub-graph which elements are fully connected with each other



Determination of connectivity in graphs

▶ **Tasks:**

- ▶ Hands-on Apache Flink / Gradoop
- ▶ Algorithms for distributed / parallel determination of **SCCs** and **Cliques**
- ▶ Development of **Gradoop operators**
- ▶ Test using example graphs
- ▶ **Evaluation** on cluster (runtime / scalability)

▶ **Technologies:**

- ▶ Apache Flink
- ▶ Gradoop

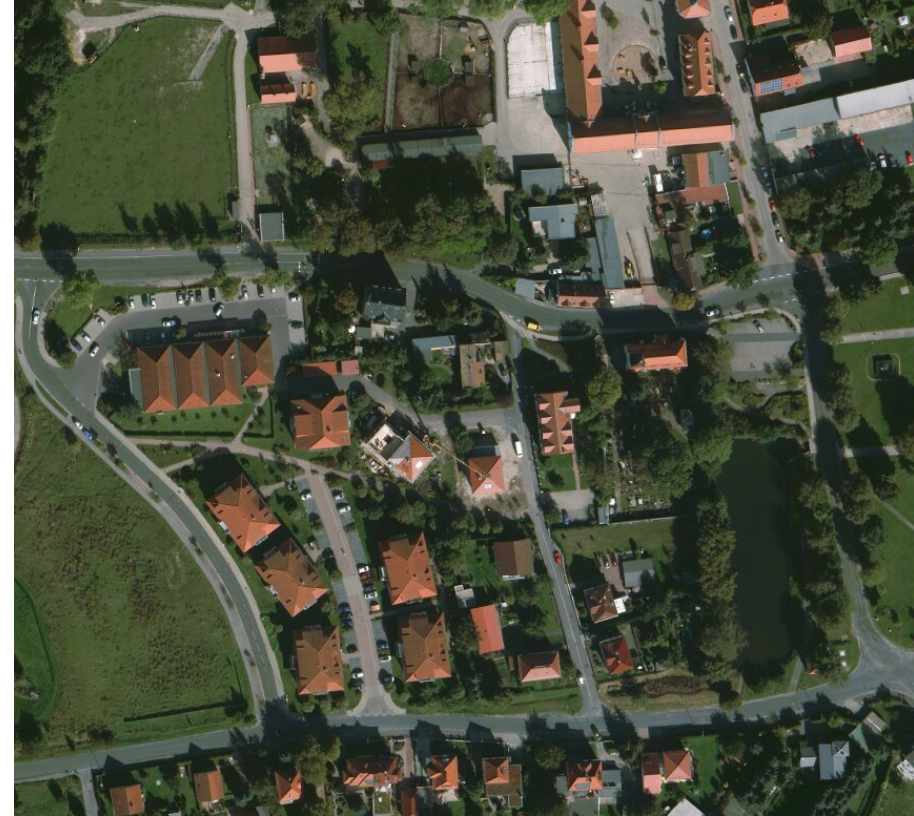
Contour Detection from Aerial Imagery for OpenStreetMap

Markus Nentwig

Motivation

Contour Detection

- ▶ Digital image feature detection
- ▶ Existing publications
 - ▶ Automatic Extraction of Building Outline [1]
 - ▶ Contour Detection with Neural Network [2]
- ▶ Existing algorithms
 - ▶ Github for [2] <https://github.com/jimeiyang/objectContourDetector>
 - ▶ Tensorflow outline detection <https://github.com/Raj-08/tensorflow-object-contour-detection>
- ▶ OpenStreetMap misses building outlines



[1] <https://pdfs.semanticscholar.org/7387/b1626abbea4205275d0b16e7bd907b54a7a2.pdf>

[2] <https://arxiv.org/pdf/1603.04530.pdf>



← original
with drawn
buildings →



← OSM map

Realization

- ▶ JOSM is a OpenStreetMap editor allowing plugins
 - ▶ Existing plugin for creating contour lines [1] as base line
- ▶ Implement your own contour detection algorithm
 - ▶ Most likely some learning-based algorithm
 - ▶ Automatic detection of colorfulness, sharpness, contrast
- ▶ JOSM Plugin (Java), TensorFlow (Python, C++)
- ▶ Expectation: get better results than base line with training
 - ▶ Perhaps even detect streets/waterways

[1] <https://github.com/JOSM/areaselector>



Tweets To Political Sentiments

Johannes Zschache



Tweets To Political Sentiments

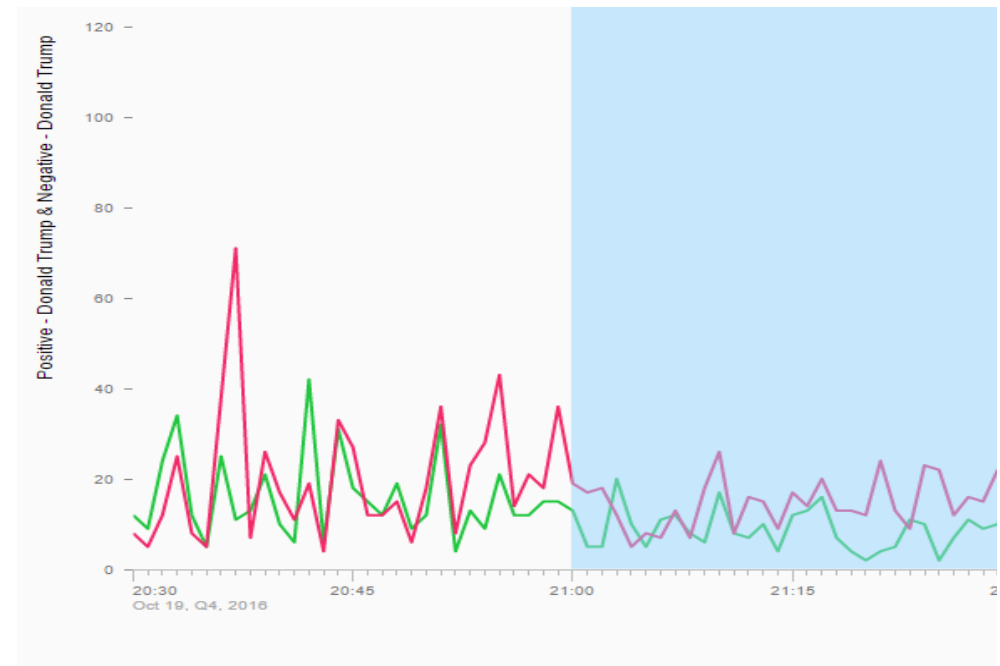
- ▶ Million people communicate on Twitter on a daily basis
 - ▶ Important source for empiric social research
 - ▶ E.g. dynamic of opinions / sentiments
- ▶ **Project:** Sentiments of Twitter users about german political parties
- ▶ Problems:
 - ▶ Filtering German language and political relation
 - ▶ Assignment of Tweets to parties (e.g. CDU, SPD, ...)
 - ▶ Sentiment Analysis
 - ▶ Bots and Opinion Spam
 - ▶ Irony

Tweets To Political Sentiments

- ▶ **Goal:** Development of a framework for collecting, filtering and cleansing of Tweets

- ▶ **Tasks:**

- ▶ Filtering of German language and political relation
- ▶ Assignment to parties
- ▶ Integration of routines (Python/R) to mark as positiv/negativ
- ▶ Connection to an NoSQL database
- ▶ Optional: Visualization



- ▶ **Technologies:** Flink Streaming/Kafka Streams, Neo4j



Traffic Analysis with Deep Learning

Moritz Wilke

Traffic Analysis with Deep Learning

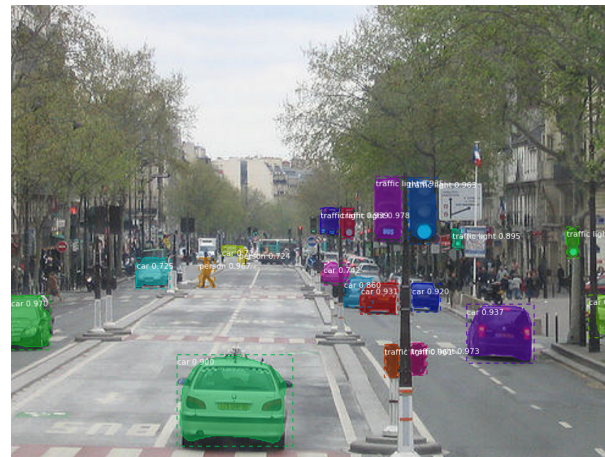
- ▶ **Goal:** Dataset and dashboard for traffic volume at Leipziger Ring
- ▶ **Tasks:**
 - ▶ Web Scraping
 - ▶ Investigation / improvement of methods for Object Recognition
 - ▶ Transformation of image data into time series
 - ▶ Data preparation for visualization, analysis or web application (dep. on interests)

Public Webcam



+

Deep Learning for Object Recognition



= ?

Integration of an ETL Process in Apache Airflow

Georges Alkhouri

Integration of an ETL Proc. in Apache Airflow

► Tasks:

- Use sensor data of the BTW 2019 DS Challenge to create an ELT Process with Airflow
- Visualize the workflow
- Scale and run the workflow on a cluster (Analyze resulting data)

- https://github.com/GeorgesAlkhouri/golddust/blob/master/golddust/data_preparation.py



Schedule	Owner	Recent Tasks [ⓘ]	Last Run [ⓘ]
00***	airflow	6	2018-09-06
*/*****	airflow	3	2018-09-05
@daily	airflow	5	2018-09-06
@once	airflow	3	2018-09-05
4:00:00	Airflow	2	2018-09-07

Security Information and Event Management

Martin Grimmer

Security Information and Event Management (SIEM)

▶ IT-term for collecting data as log-files in a central repo. and the subsequent analysis

▶ SIEM contains:

▶ Software agents

- ▶ Run on computer system that should be monitored
- ▶ Send log information to central server

▶ Analytics server

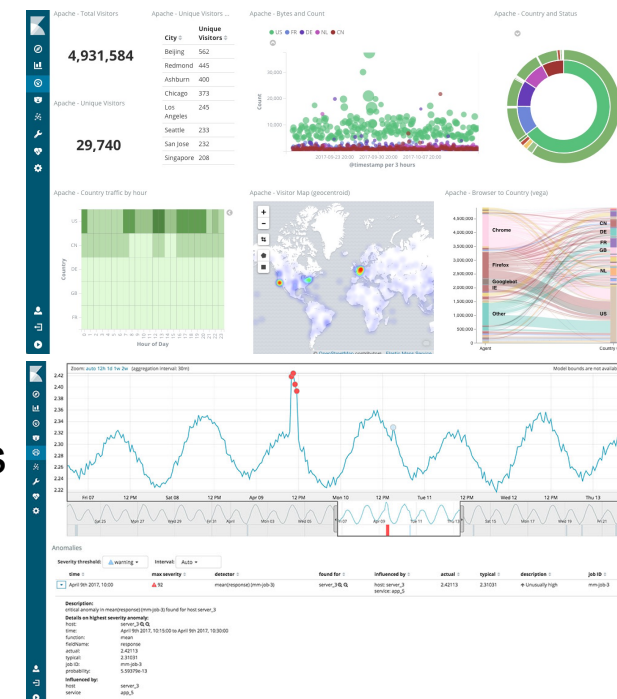
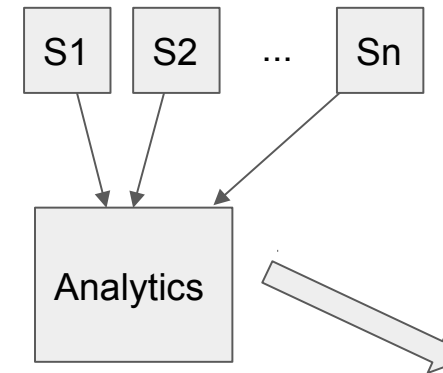
- ▶ Store log data in queryable form for analysis
- ▶ Creates reports and diagrams in real-time

▶ Tasks:

- ▶ Implement prototype of a SIEM
- ▶ Collect data, run simple analysis tasks and visualize results

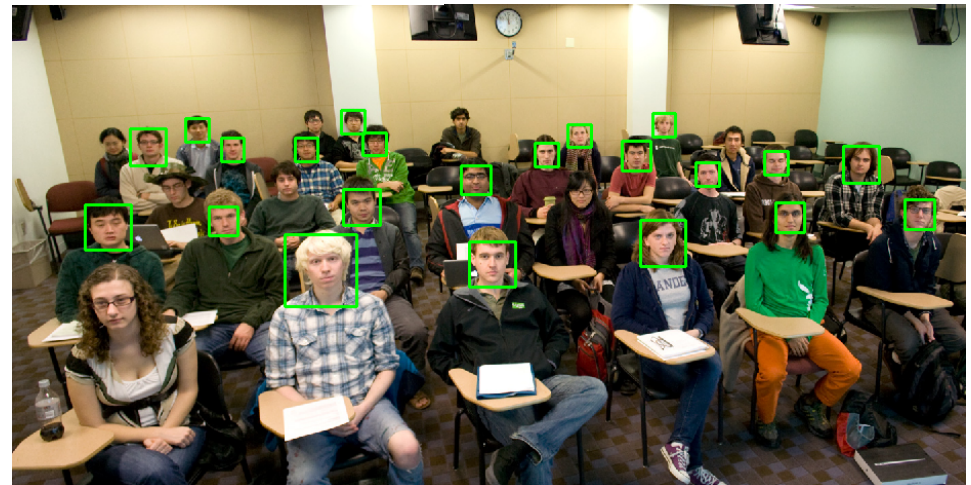
▶ Technologies:

- ▶ Docker, Sysdig, ElasticSearch, Kibana, ...



Face Recognition for Entry Assistants

- ▶ Evaluation of different methods (feature-based or geometric) for
 - ▶ Face detection
 - ▶ Face recognition
 - ▶ Emotion recognition
- ▶ Integration in Harambe 2.0 prototype from SWT practical course



Themenübersicht

#	Thema	Technologie	Betreuer
1	(Incremental) PPRL on Flink	Apache Flink, Java	Franke
2	Cryptanalysis for Bloom-Filter-based PPRL	Apache Flink	Rohde
3	Parallel Multi-Party PPRL	Apache Flink	Sehili
4	Rollup Querying of Ontology Graphs	Apache Flink, Gradoop	Christen
5	Processing and Visualization of Graph Streams	Apache Flink	Rost
6	Distributed Graph Sampling using Forest Fire	Apache Flink / Gelly / Pregel / Gradoop	Gomez
7	Representation Learning for Gradoop	Apache Flink, Gradoop	Obraczka
8	Determination of connectivity in graphs	Apache Flink / Gradoop	Täschner
9	Contour Detection from Aerial Imagery for OpenStreetMap	JOSM Plugin (Java), TensorFlow (Python,C++)	Nentwig
10	Tweets To Political Sentiments	Apache Flink Streaming/Kafka Streams, Neo4j	Zschache
11	Traffic Analysis with Deep Learning		Wilke
12	Integration of an ETL Process in Apache Airflow	Apache Airflow	Alkhouri
13	Security Information and Event Management	Docker, Sysdig, ElasticSearch	Grimmer
14	Face Recognition for Entry Assistants		Peukert