UNIVERSITÄT BONN

SMART DATA ANALYTICS
FROM DATA TO KNOWLEDGE

MA-INF 4223 - Lab Distributed Big Data Analytics

Spark Fundamentals II

Dr. Hajira Jabeen, Gezim Sejdiu

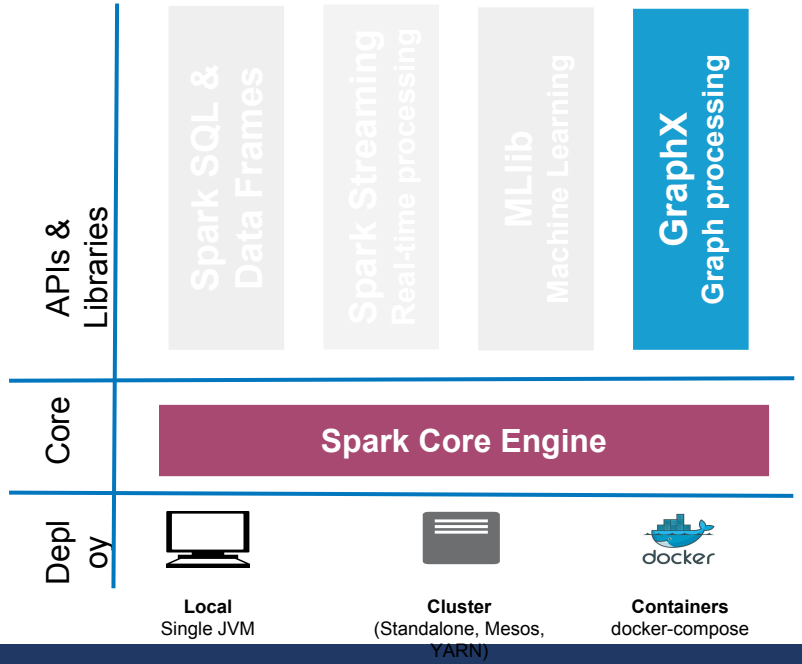
Winter Semester 2017/18

Lesson objectives

- After completing this lesson, you should be able to:
  - Understand and use various Spark Libraries
    - Spark GraphX - graph processing
    - Spark MLlib

Spark GraphX

Spark GraphX

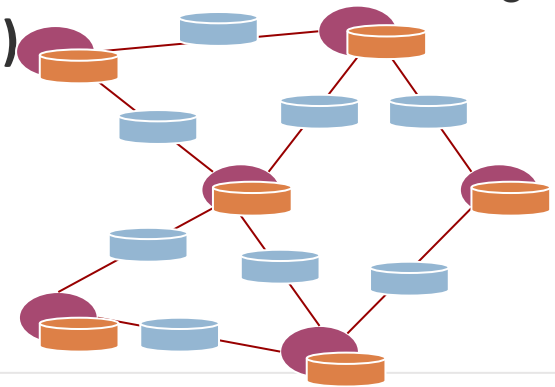


Spark GraphX

- Graph computation system which runs in the Spark data-parallel framework.
- GraphX extends Spark's Resilient Distributed Dataset (RDD) abstraction to introduce the Resilient Distributed Graph (RDG)

Spark GraphX

- Spark GraphX - stands for graph processing
  - For graph and graph-parallel computation
- At a high level, GraphX extends the Spark RDD by introducing a new Graph abstraction:
  - a directed multigraph with properties attached to each vertex and edge.
- It is based on Property Graph model → **G(V, E)**
  - Vertex Property
    - Triple details
  - Edge Property
    - Relations
    - Weights



Resilient Distribute Graph (RDG)

- A tabular representation of the efficient vertex-cut partitioning and data-parallel partitioning heuristics
- Supports implementations of the
  - PowerGraph and
  - Pregel graph-parallel
- Preliminary performance comparisons between a popular dataparallel and graph-parallel frameworks running PageRank on a large real-world graph

Graph Parallel

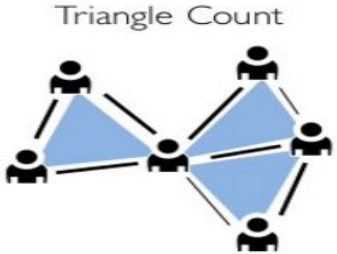
- Graph-parallel computation typically adopts a vertex (and occasionally edge) centric view of computation
- Retaining the **data-parallel metaphor,** program logic in the GraphX system defines transformations on graphs with each operation yielding a new graph
- The core data-structure in the GraphX systems is an immutable graph

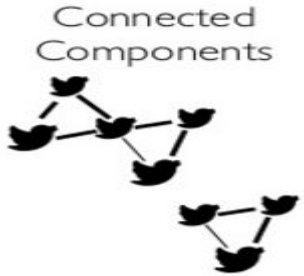
GraphX operations

```
class Graph[VD, ED] {
// Information about the Graph
val numEdges: Long
val numVertices: Long
val inDegrees: VertexRDD[Int]
val outDegrees: VertexRDD[Int]
val degrees: VertexRDD[Int]

// Views of the graph as collections
val vertices: VertexRDD[VD]
val edges: EdgeRDD[ED]
val triplets: RDD[EdgeTriplet[VD, ED]]

// Functions for caching graphs
def persist(newLevel: StorageLevel = StorageLevel.MEMORY_ONLY): Graph[VD, ED]
def cache(): Graph[VD, ED]
def unpersistVertices(blocking: Boolean = true): Graph[VD, ED]
// Change the partitioning heuristic
def partitionBy(partitionStrategy: PartitionStrategy): Graph[VD, ED]
// Transform vertex and edge attributes
def mapVertices[VD2](map: (VertexId, VD) => VD2): Graph[VD2, ED]
def mapEdges[ED2](map: Edge[ED] => ED2): Graph[VD, ED2]
        ----
```

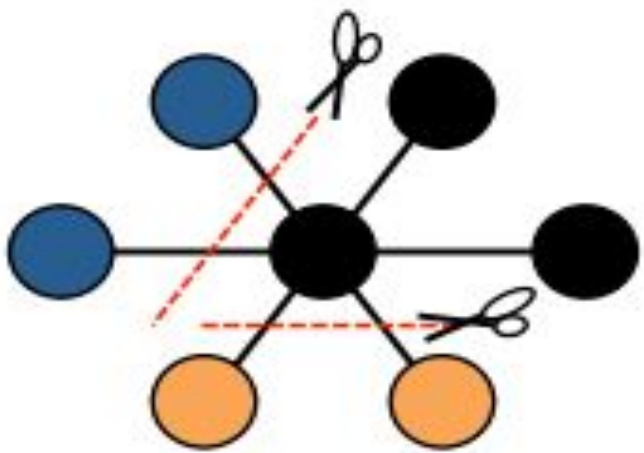
GraphX build-in Graph Algorithms

```
// Basic graph algorithms
================================================================
def pageRank(tol: Double, resetProb: Double = 0.15): Graph[Double, Double]
def connectedComponents(): Graph[VertexId, ED]
def triangleCount(): Graph[Int, ED]
def stronglyConnectedComponents(numIter: Int): Graph[VertexId, ED]
```

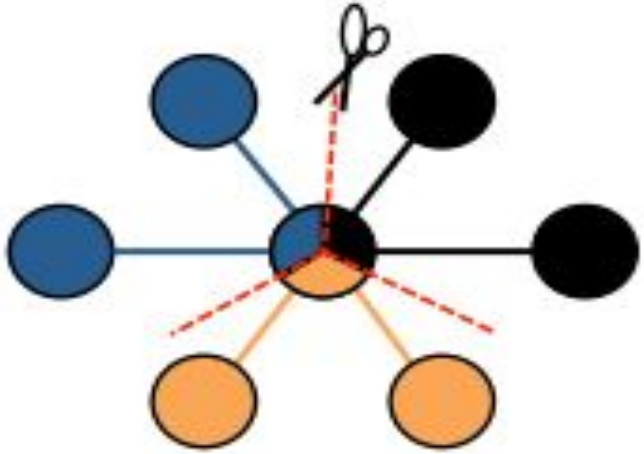




Edge-Cut vs Vertex-Cut

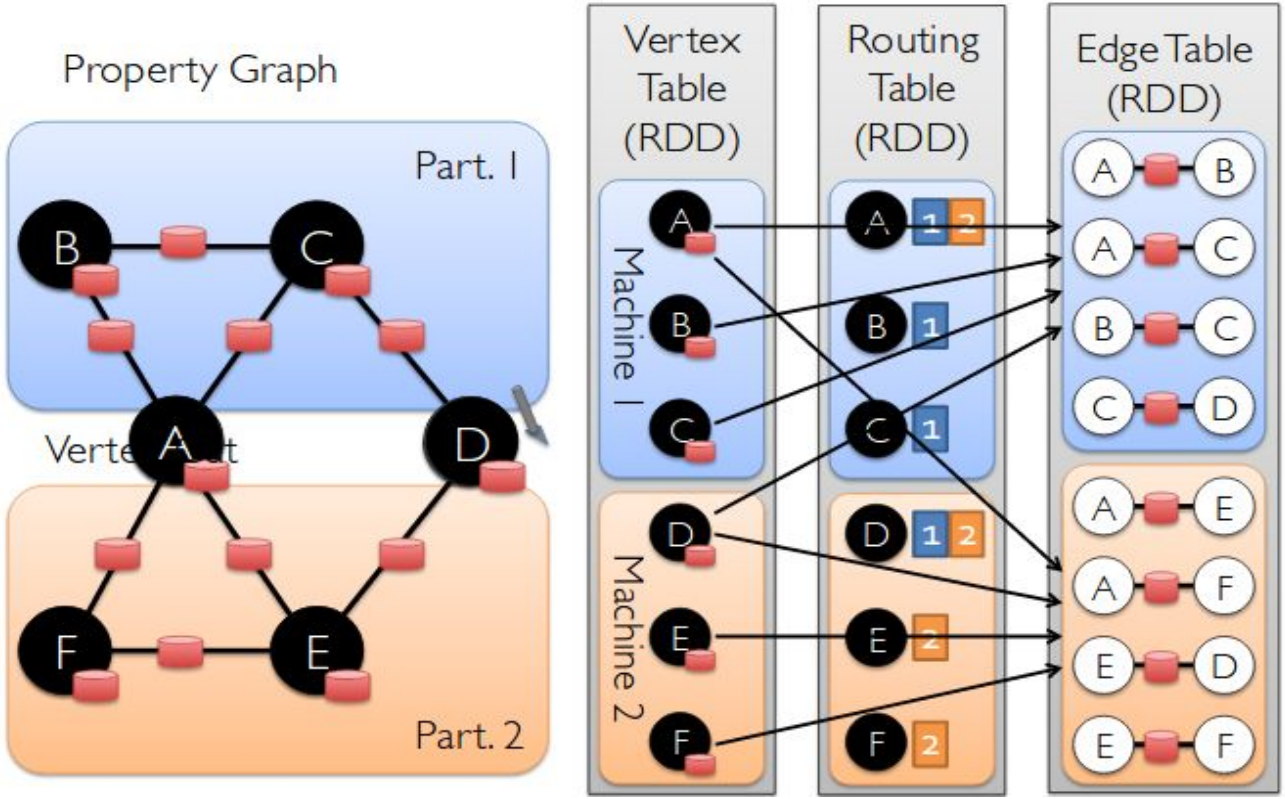


(a) Edge-Cut



(b) Vertex-Cut

Encoding Property Graphs as RDDs



Edge Table

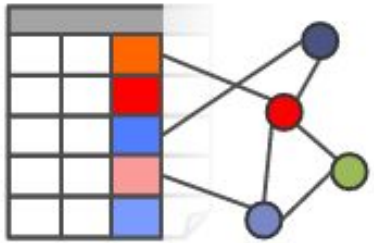
- EdgeTable(pid, src, dst, data): stores the adjacency structure and edge data
- Each edge is represented as a tuple consisting of the
  - source vertex id,
  - destination vertex id,
  - user-defined data
  - virtual partition identifier (pid).

Vertex Data Table

- VertexDataTable(id, data): stores the vertex data, in the form of a vertex (id, data) pairs
- VertexMap(id, pid): provides a mapping from the id of a vertex to the ids of the virtual partitions that contain adjacent edges

New API

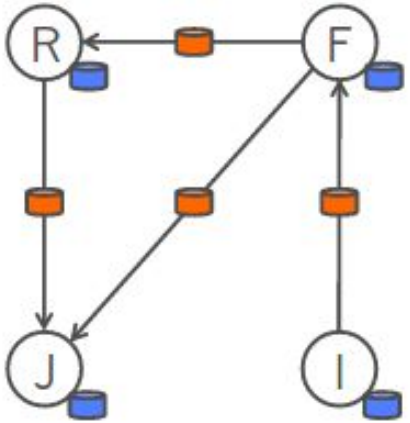
Blurs the distinction between Tables and Graphs



New Library

Embeds Graph-Parallel model in Spark

Property Graph



Vertex Table

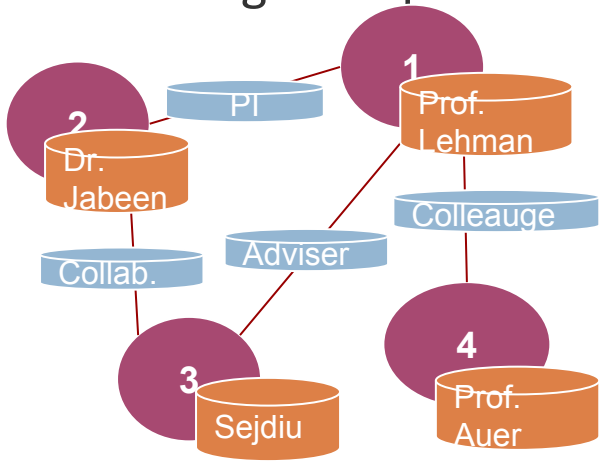
Id	Attribute (V)
Rxin	(Stu., Berk.)
Jegonzal	(PstDoc, Berk.)
Franklin	(Prof., Berk)
Istoica	(Prof., Berk)

Edge Table

SrcId	DstId	Attribute (E)
rxin	jegonzal	Friend
franklin	rxin	Advisor
istoica	franklin	Coworker
franklin	jegonzal	PI

Spark GraphX - Getting Started

Creating a Graph




```
type VertexId = Long
// Create an RDD for the vertices
val users: RDD[(VertexId, (String, String))]=
spark.sparkContext.parallelize(
                Array((3L, ("sejdiu", "phd_student")),
                      (2L, ("jabeen", "postdoc")),
                      (1L, ("lehmann", "prof")),
                      (4L, ("auer", "prof"))))
// Create an RDD for edges
val relationships: RDD[Edge[String]] =
spark.sparkContext.parallelize(
                Array(Edge(3L, 2L, "collab"),
                      Edge(1L, 3L, "advisor"),
                      Edge(1L, 4L, "colleague"),
                      Edge(2L, 1L, "pi")))
// Build the initial Graph
val graph = Graph(users, relationships)
```

Vertex RDD

vID	Property(V)
1L	(lehmann, prof)
2L	(jabenn, postdoc)
3L	(sejdiu, phd_student)
4L	(auer, prof)

Edge RDD

sID	dID	Property(E)
1L	3L	advisor
1L	4L	colleague
2L	1L	pi
3L	2L	collab

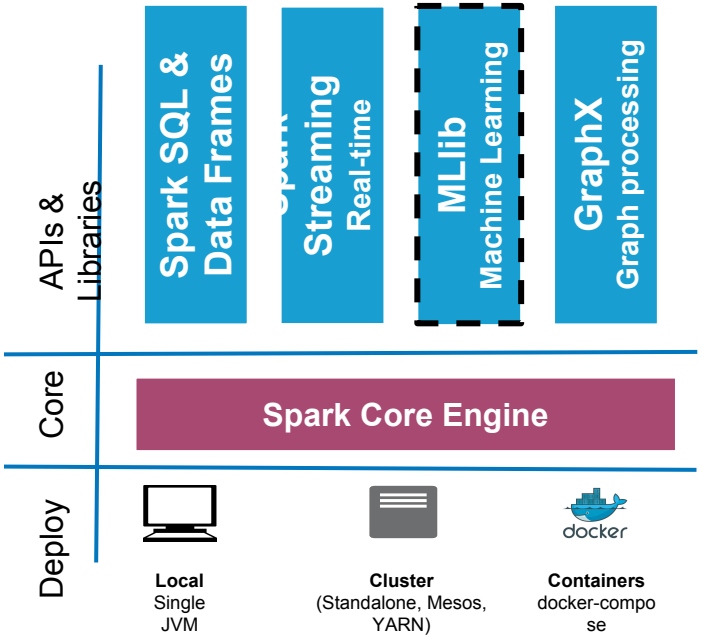
GraphX Optimizations

- Mirror Vertices
- Multicast Joins
- Partial materialization
- Incremental view
- Index Scanning for Active Sets
- Local Vertex and Edge Indices
- Index and Routing Table Reuse

Spark ML

MLlib

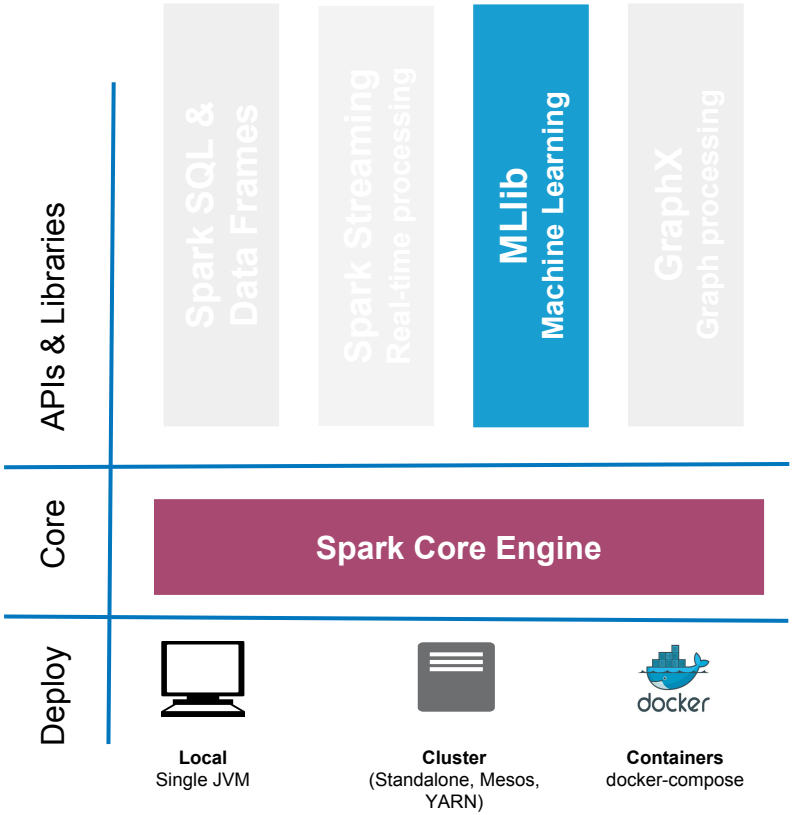
- Unified stack



Overview

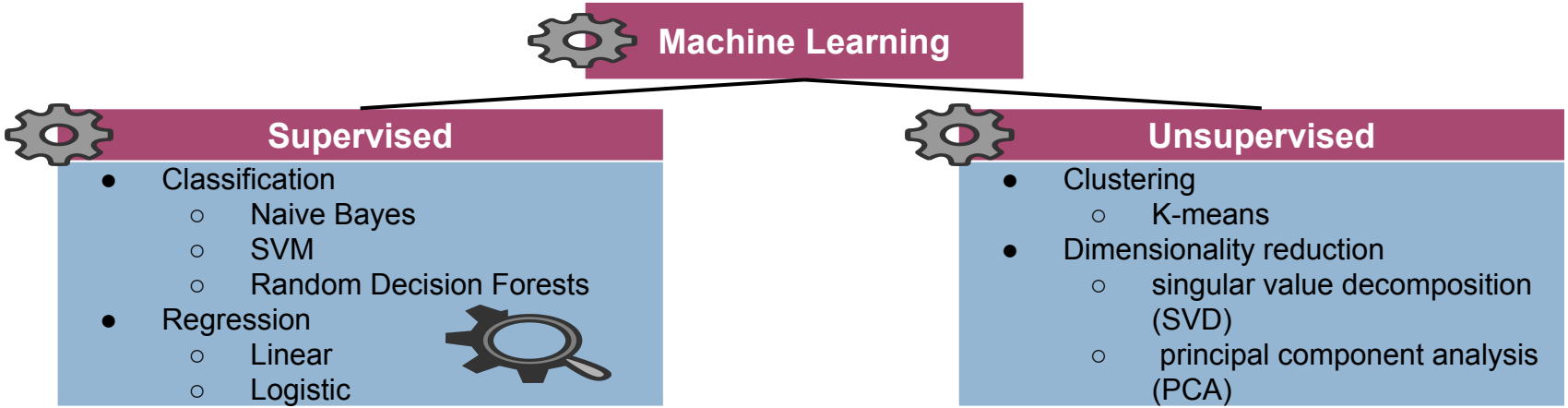
- MLlib: Machine Learning in Apache Spark

Spark ML


APIs & Libraries
Spark SQL & Data Frames
Spark Streaming
Real-time processing
MLlib
Machine Learning
GraphX
Graph processing
Core
Spark Core Engine
Deploy
Local
Single JVM
Cluster
(Standalone, Mesos, YARN)
Containers
docker-compose
docker

ML Algorithms overview

- Machine learning are separated in two major types of algorithms :
  - Supervised - labeled data in which both, input and output are provided to the algorithm
  - Unsupervised - do not have the outputs in advance



Spark ML

- MLlib is a standard component of Spark providing machine learning primitives on top of Spark
- It is scalable machine learning, statistics, math libraries
- Supports out-of-the-box most popular machine learning algorithms like Linear regression, Logistic regression, Decision Trees
- Is available in Scala, Java, Python, and R

Spark ML-pipelines

- Uniform set of APIs for creating and tuning data processing/machine learning pipelines
- Core concepts:
  - DataFrame: RDD with names columns. SQL-like syntax and other core RDD operations
  - Transformer: DataFrame => DataFrame. Eg., features to predictions(classifier)
  - Estimator: DataFrame => Transformer. e.g., learning algorithm
  - Param: map of params
  - Pipeline: Chain of Transformers and Estimators. Specifies the data flow

Spark ML-pipelines

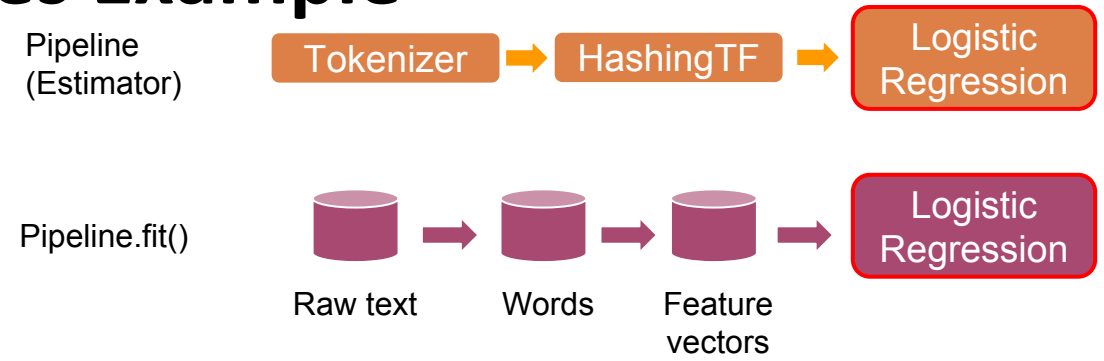
- Transformer
  - A Transformer is an abstraction which uses an algorithm to transform one DataFrame to another
  - It implements a method `transform()`

Spark ML-pipelines

- Estimator
  - An Estimator abstraction uses an algorithm which is fitted into a DataFrame returning a model
  - It implements a method `fit()`

Spark ML-pipelines Example

- Split text into words => convert numerical features => generate a prediction model




```
val tokenizer = new Tokenizer().setInputCol("text").setOutputCol("words")
val hashingTF = new HashingTF().setNumFeatures(1000).setInputCol(tokenizer.getOutputCol)
.setOutputCol("features")
val lr = new LogisticRegression().setMaxIter(10).setRegParam(0.01)
val pipeline = new Pipeline().setStages(Array(tokenizer, hashingTF, lr))
val model = pipeline.fit(training.toDF)
val test = sc.parallelize(Seq(
Document(4L, "spark i j k"),
Document(5L, "l m n"),
Document(6L, "mapreduce spark"),
Document(7L, "apache hadoop")))
val predictions = model.transform(test.toDF)
```

References

[1]. GraphX: Graph Processing in a Distributed Dataflow Framework by Gonzalez, Joseph, Reynold Xin, Ankur Dave, Daniel Crankshaw, Michael J. Franklin and Ion Stoica *in OSDI,* 2014.

[2]. “GraphX Programming Guide” - http://spark.apache.org/docs/latest/graphx-programming-guide.html

[3]. MLlib: Machine Learning in Apache Spark by Meng, Xiangrui, Joseph K. Bradley, Burak Yavuz, Evan R. Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, D. B. Tsai, Manish Amde, Sean Owen, Doris Xin, Reynold Xin, Michael J. Franklin, Reza Bosagh Zadeh, Matei Zaharia and Ameet Talwalkar *in Journal of Machine Learning Research 17*, 2016.

[4]. “Machine Learning Library (MLlib) Guide” - http://spark.apache.org/docs/latest/ml-guide.html

THANK YOU !

http://sda.cs.uni-bonn.de/teaching/dbda/

- http://sda.cs.uni-bonn.de/
- https://github.com/SANSA-Stack
- https://github.com/big-data-europe
- https://github.com/SmartDataAnalytics

Dr. Hajira Jabeen
jabeen@cs.uni-bonn.de
Room A108 (Appointment per e-mail)

Gezim Sejdiu
sejdiu@cs.uni-bonn.de
Room A120 (Appointment per e-mail)