SANSA: Distributed Semantic Analytics

LEARNING, APPLYING, MULTIPLYING BIG DATA ANALYTICS

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under grant agreement No 809965.

Killing two birds with one stone…

Why not installing SANSA during the presentation? ☺

- From a the SANSA-Notebooks github repository <https://github.com/sansa-stack/sansa-notebooks>

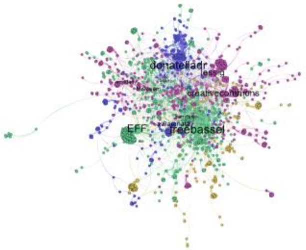
- Requirements:
  - docker
  - docker-compose

- Only **5** commands:
  - `git clone https://github.com/sansa-stack/sansa-notebooks`
  - `cd sansa-notebooks/`
  - `make`
  - `make up`
  - `make load-data`
  - Goto: http://localhost/

Learning, Applying, Multiplying Big Data Analytics

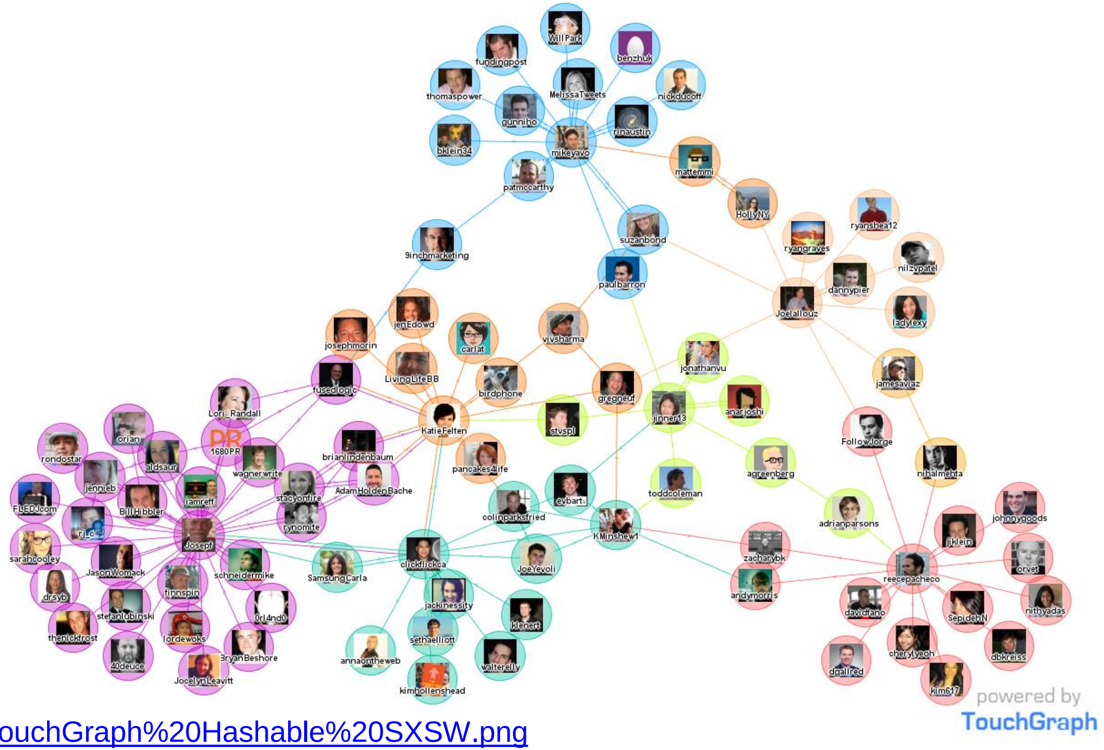
SANSA: Motivation

- Abundant machine readable structured information is available (e.g. in RDF)
  - Across SCs, e.g. Life Science Data
  - General: DBpedia, Google knowledge graph
  - Social graphs: Facebook, Twitter
- Need for scalable querying, inference and machine learning
  - Link prediction
  - Knowledge base completion
  - Predictive analytics



Social Networks

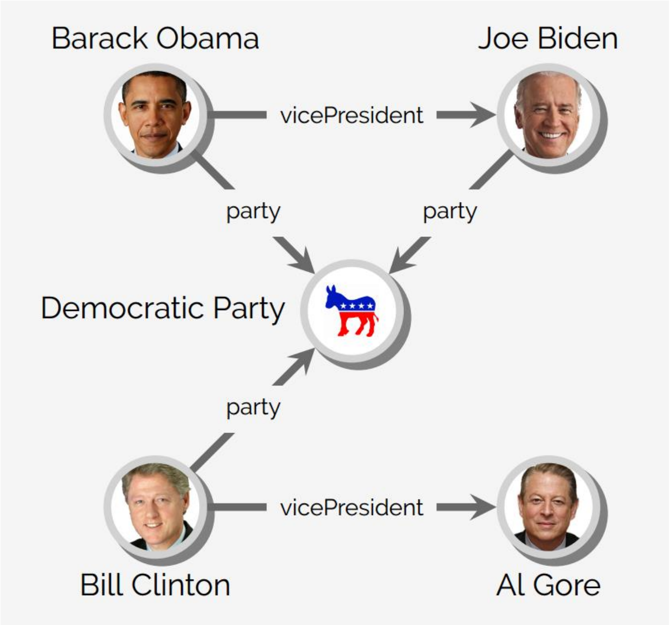
Social Networks as Graphs



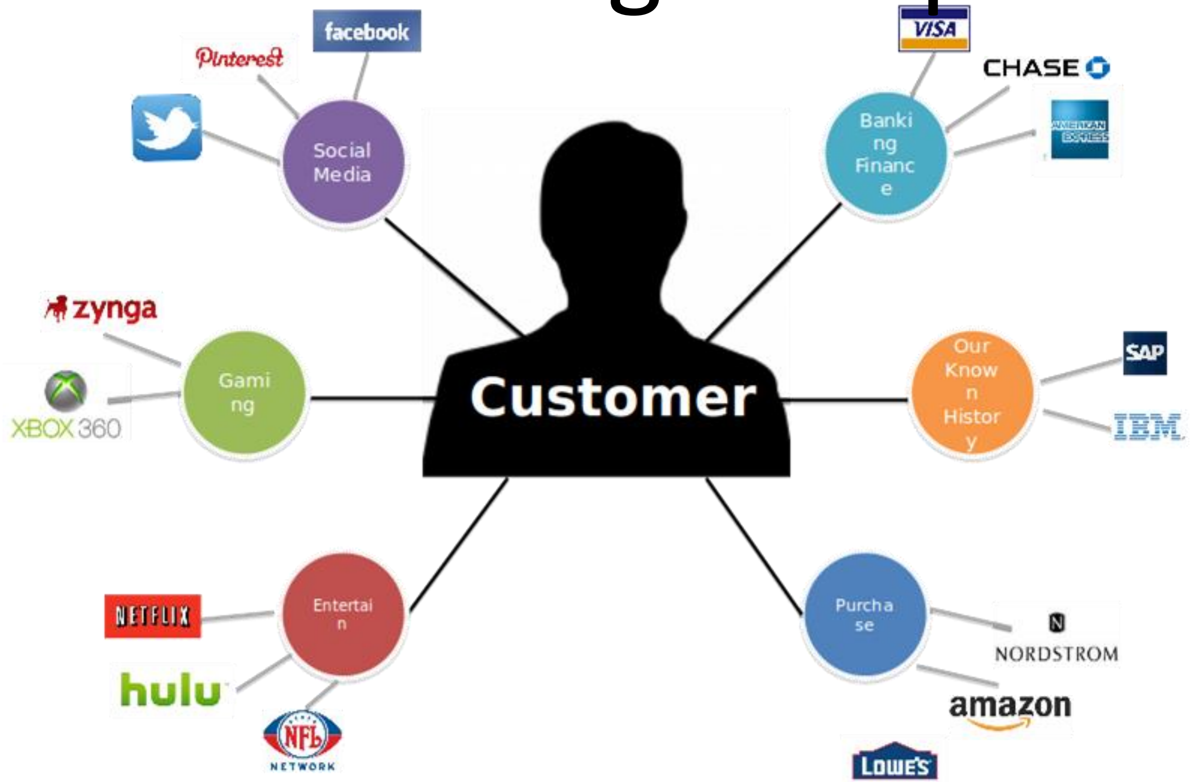
http://www.touchgraph.com/assets/images/TouchGraph%20Hashable%20SXSW.png

Knowledge Graphs

- Modelling entities and their relationships
- Analysis: finding underlying structure of graph e.g. to predict unknown relationships
- Examples: Google Knowledge Graph, DBpedia, Facebook, YAGO, Twitter, LinkedIn, MS Academic Graph, WikiData

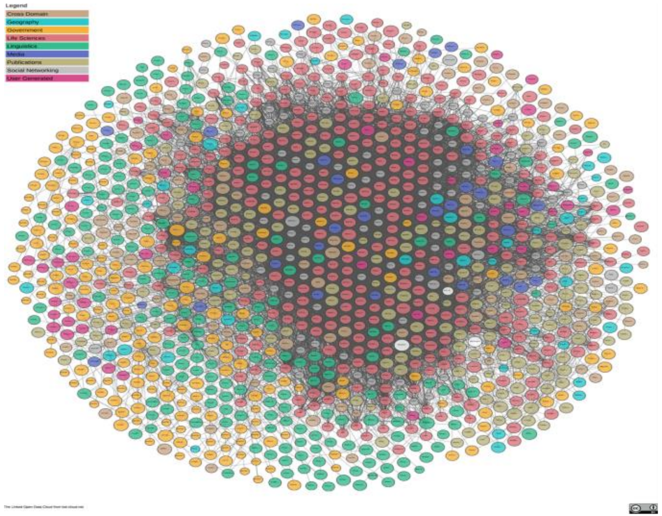


Knowledge Graph



Motivation

- Over the last years, the size of the Semantic Web has increased and several large-scale datasets were published
  - As of August 2018



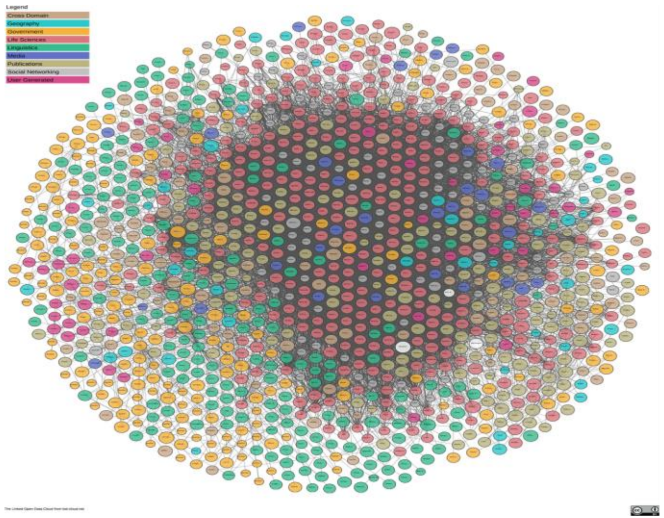
Source: LOD-Cloud (http://lod-cloud.net/)

Motivation

- Over the last years, the size of the Semantic Web has increased and several large-scale datasets were published
  - Based on LOD Stats (http://lodstats.aksw.org/)

~10, 000 datasets

Openly available online using Semantic Web standards

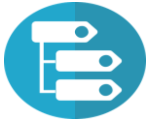


Source: LOD-Cloud (http://lod-cloud.net/)

Motivation

❖ Over the last years, the size of the Semantic Web has increased and several large-scale datasets were published

➢ Based on LOD Stats (http://lodstats.aksw.org/)

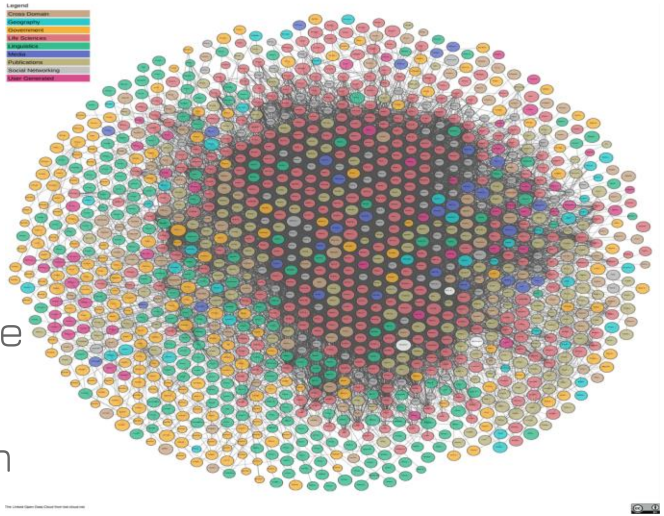


~10, 000 datasets

Openly available online using Semantic Web standards

many datasets

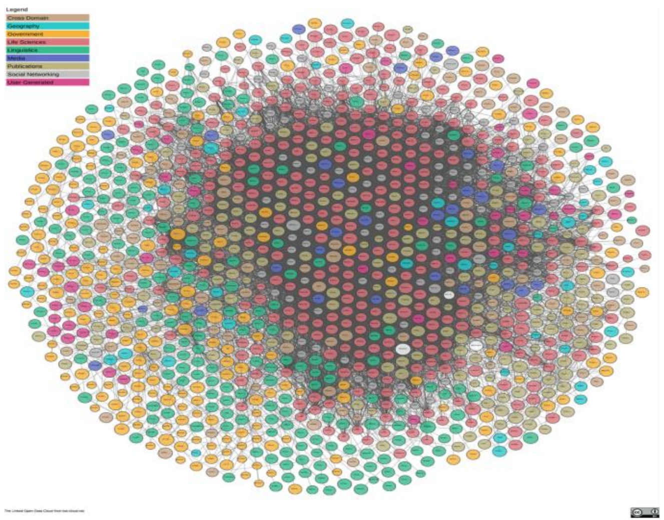
RDFized and kept private (e.g. Supply chain, manufacture, ethereum dataset, etc.)



Source: LOD-Cloud (http://lod-cloud.net/)

Motivation

- Dealing with such amount of data makes many tasks hard to be solved on single machines



Source: LOD-Cloud (http://lod-cloud.net/)

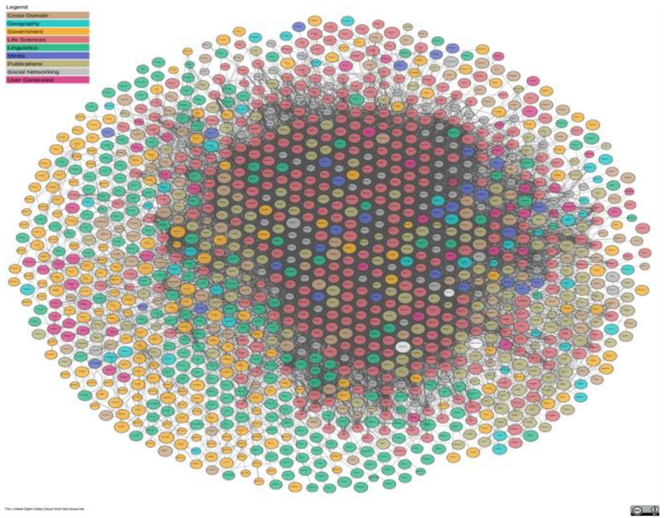
Learning, Applying, Multiplying Big Data Analytics

Motivation

❖ Dealing with such amount of data makes many tasks hard to be solved on single machines

Vocabulary Reuse

Find a suitable vocabulary for your dataset



Source: LOD-Cloud (http://lod-cloud.net/)

Motivation

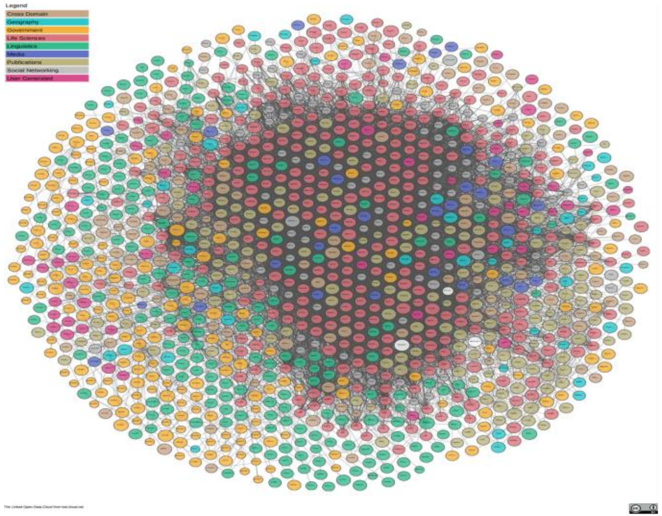
❖ Dealing with such amount of data makes many tasks hard to be solved on single machines

Vocabulary Reuse

Find a suitable vocabulary for your dataset

Coverage Analysis

Does dataset contain necessary information?



Source: LOD-Cloud (http://lod-cloud.net/)

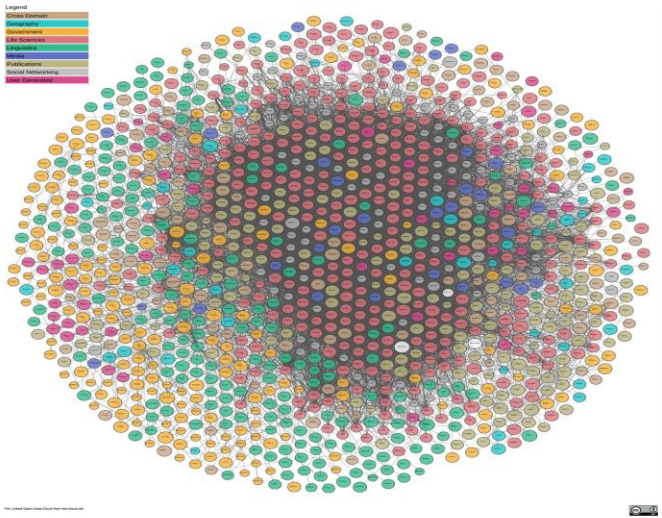
Motivation

❖ Dealing with such amount of data makes many tasks hard to be solved on single machines

Vocabulary Reuse
Find a suitable vocabulary for your dataset

Coverage Analysis
Does dataset contain necessary information?

Privacy Analysis
Does dataset contain sensitive information?



Source: LOD-Cloud (http://lod-cloud.net/)

Motivation

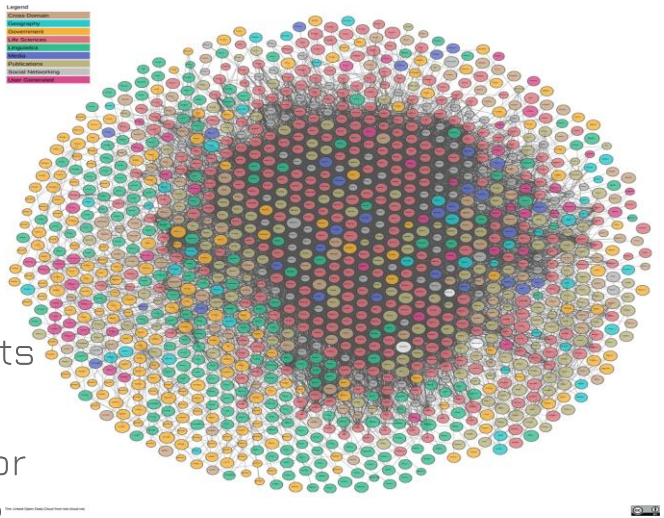
- Dealing with such amount of data makes many tasks hard to be solved on single machines

Vocabulary Reuse
Find a suitable vocabulary for your dataset

Coverage Analysis
Does dataset contain necessary information?

Privacy Analysis
Does dataset contain sensitive information?

Entity Linking
Which datasets are good candidates for interlinking?



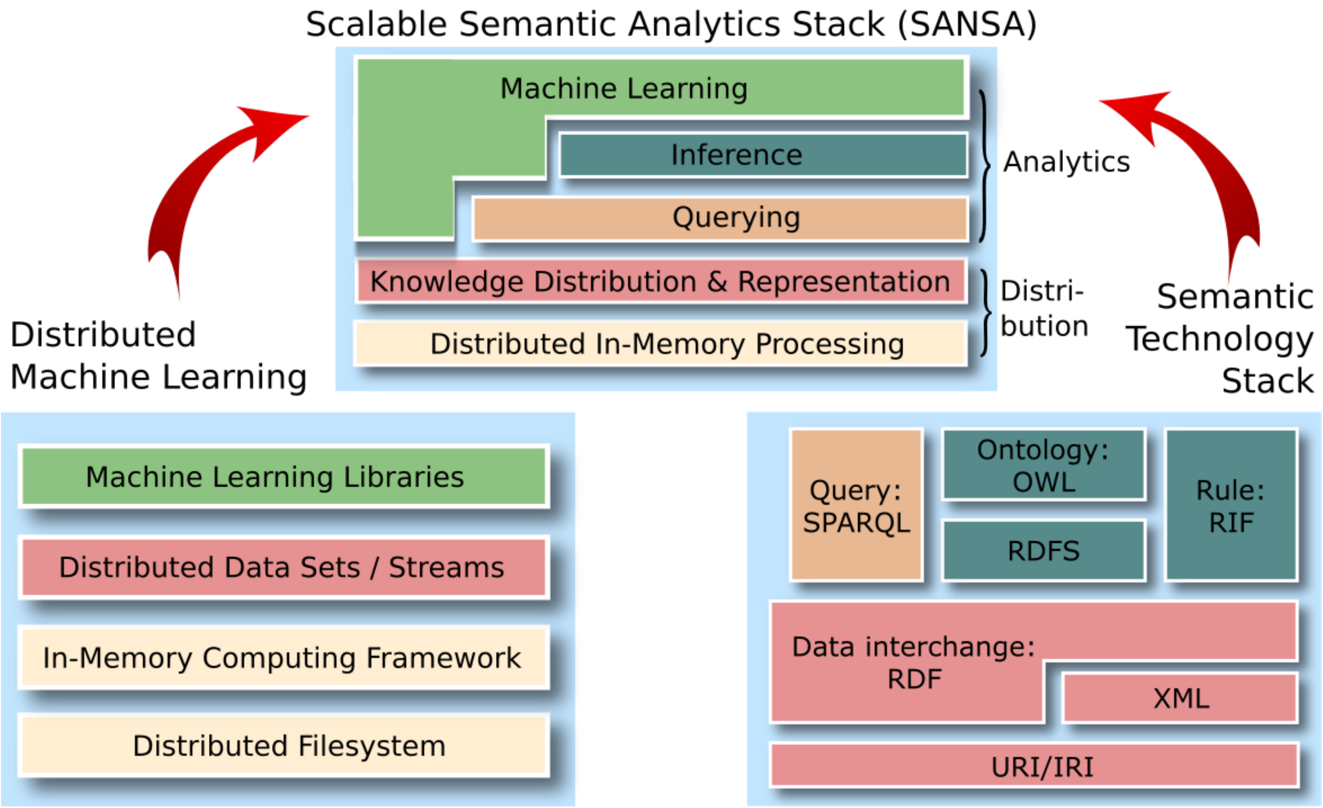
Source: LOD-Cloud (http://lod-cloud.net/)

Why Distributed RDF Data Processing?

Tasks hard to solve on single machines (>1TB memory consumption):

- **Querying** and processing LinkedGeoData
- Dataset statistics and **quality assessment** of the LOD Cloud
- Vandalism and **outlier detection** in Wikidata
- **Inference** on life science data (e.g. UniProt, EggNOG, StringDB)
- Clustering of DBpedia data
- **Clustering** of user-logs of the Big Data Europe integrator platform for the creation of user profiles
- Large-scale enrichment and **link prediction** for e.g. DBpedia → LinkedGeoData

SANSA Stack Vision



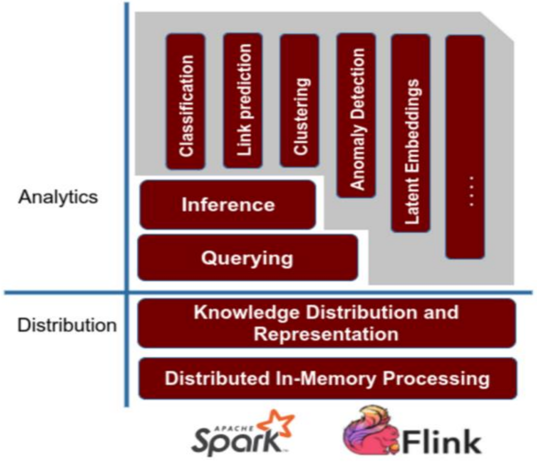
Why combining Big Data and SW?

	"Big Data" Processing (Spark/Flink)	Semantic Technology Stack
Data Integration	Manual pre-processing	Partially automated, standardised
Modelling	Simple (often flat feature vectors)	Expressive
Support for data exchange	Limited (heterogeneous formats with limited schema information)	Yes (RDF & OWL W3C Recommendations)
Business value	Direct	Indirect
Horizontally scalable	Yes	No

Idea: combine advantages of both worlds

SANSA Stack

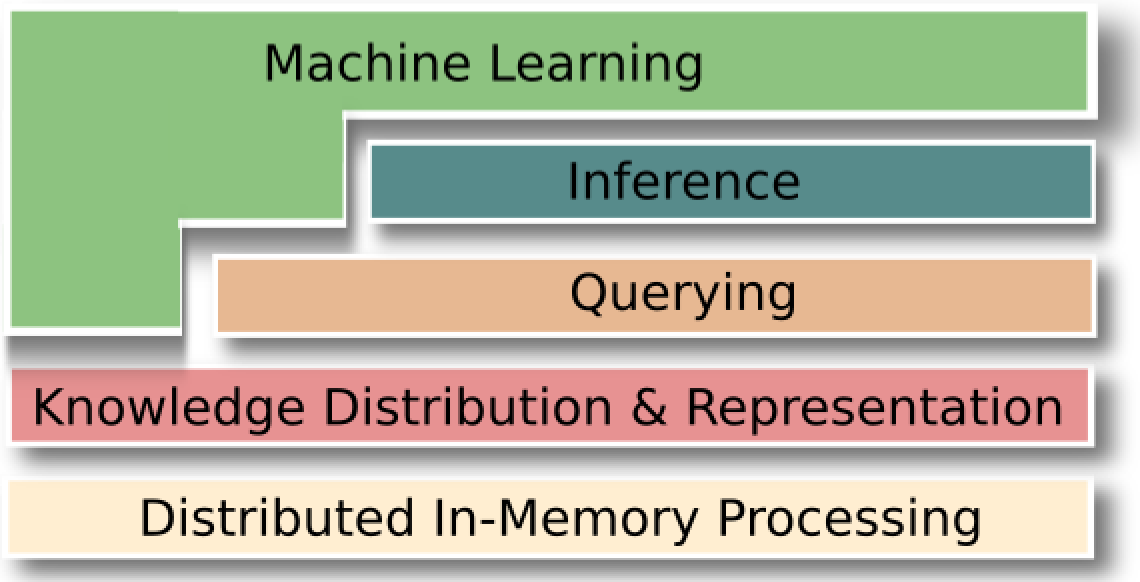
- It's core is a **processing data flow** engine that provides data distribution, and fault tolerance for distributed computations over RDF large-scale datasets
- SANSA includes **several libraries** for creati
  - Read / Write RDF / OWL library
  - Querying library
  - Inference library
  - ML- Machine Learning core library



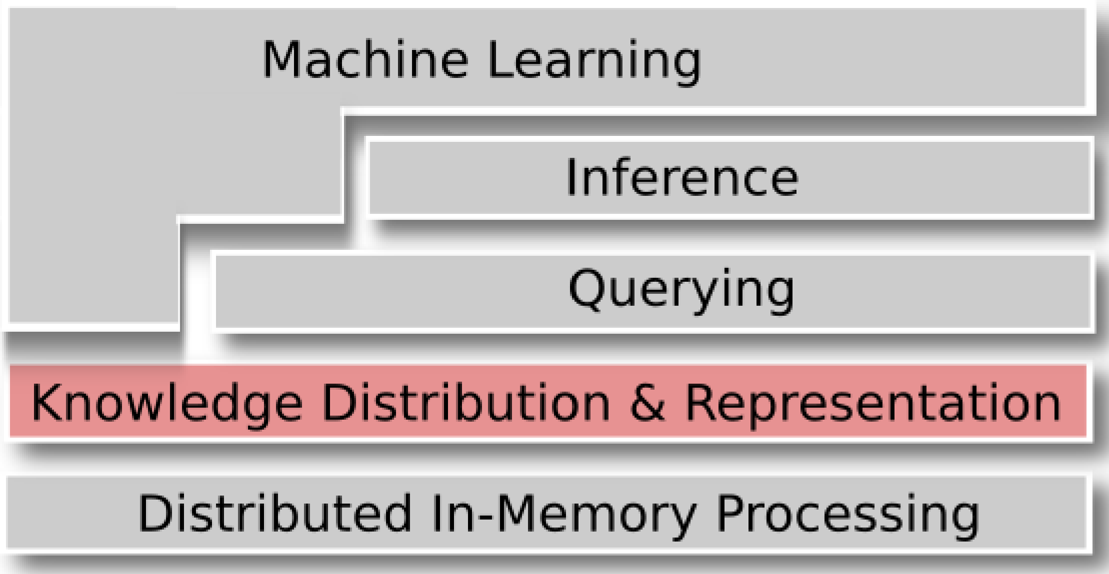
http://sansa-stack.net/

LEARNING, APPLYING, MULTIPLYING BIG DATA ANALYTICS

SANSA Layers



SANSA: Read Write Layer

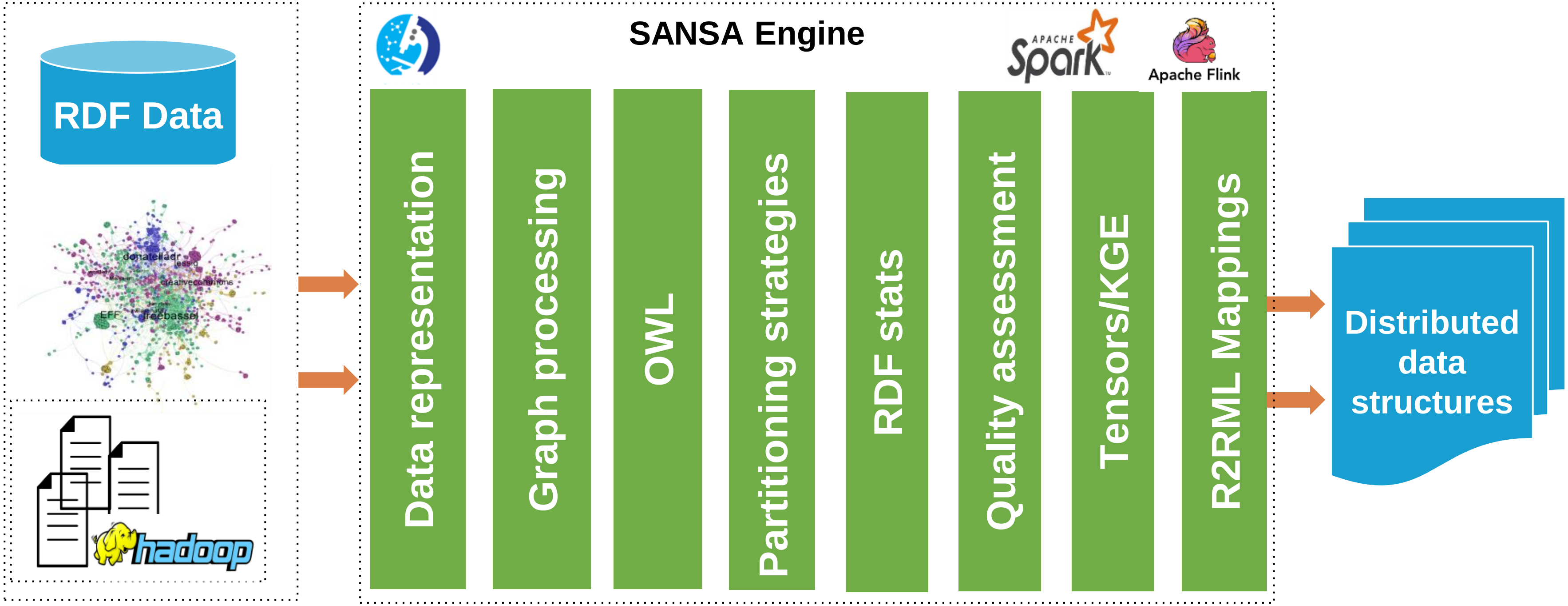


SANSA: Read Write Layer

- ❖ Ingest RDF and OWL data in different formats using Jena / OWL API style interfaces
- ❖ Represent data in multiple formats
  - ➢ (e.g. RDD, Data Frames, GraphX, Tensors)
- ❖ Allow transformation among these formats
- ❖ Compute dataset statistics and apply functions to URIs, literals, subjects, objects → Distributed LODStats

```
val triples = spark.rdf(Lang.NTRIPLES)(input)
triples.find(None,
Some(NodeFactory.createURI("http://dbpedia.org/ontology/influenced")), None)
val rdf_stats_prop_dist = triples.statsPropertyUsage()
```

SANSA: Read Write Layer features



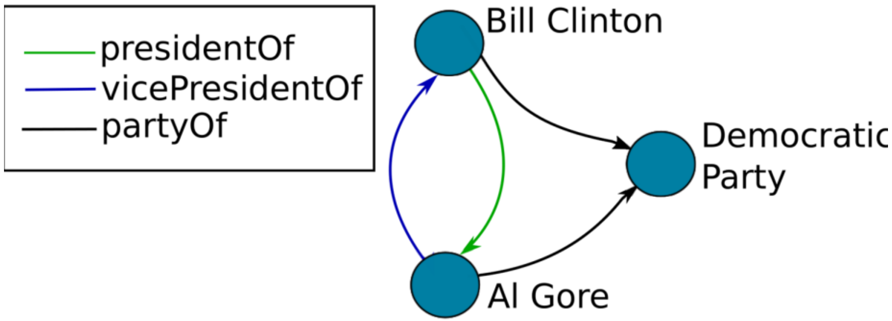
SANSA: OWL Support

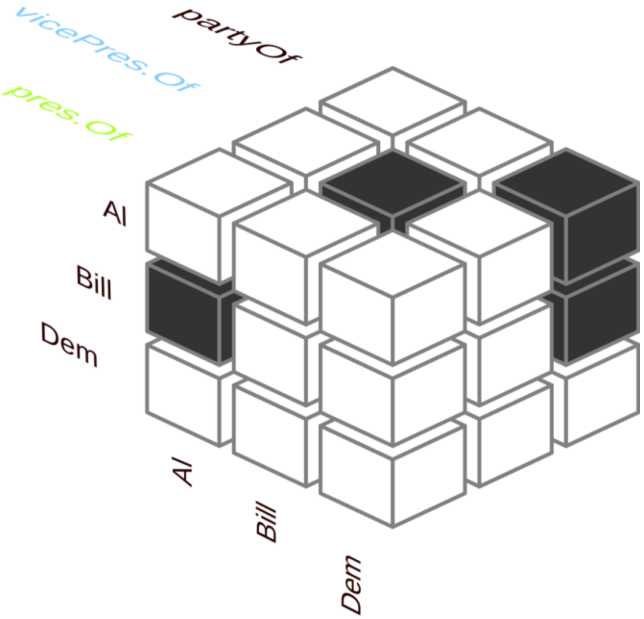
- Distributed processing of OWL axioms
- Support for Manchester OWL & functional syntax
- Derived distributed data structures:
  - E.g. matrix representation of subclass-of axioms to compute its closure via matrix operations

```
val rdd =spark.owl(Syntax.MANCHESTER)("file.owl")
// get all subclass-of axioms
val sco = rdd.filter(_.isInstanceOf[OWLSubClassOfAxiom])
```

RDF to Tensors (experimental)

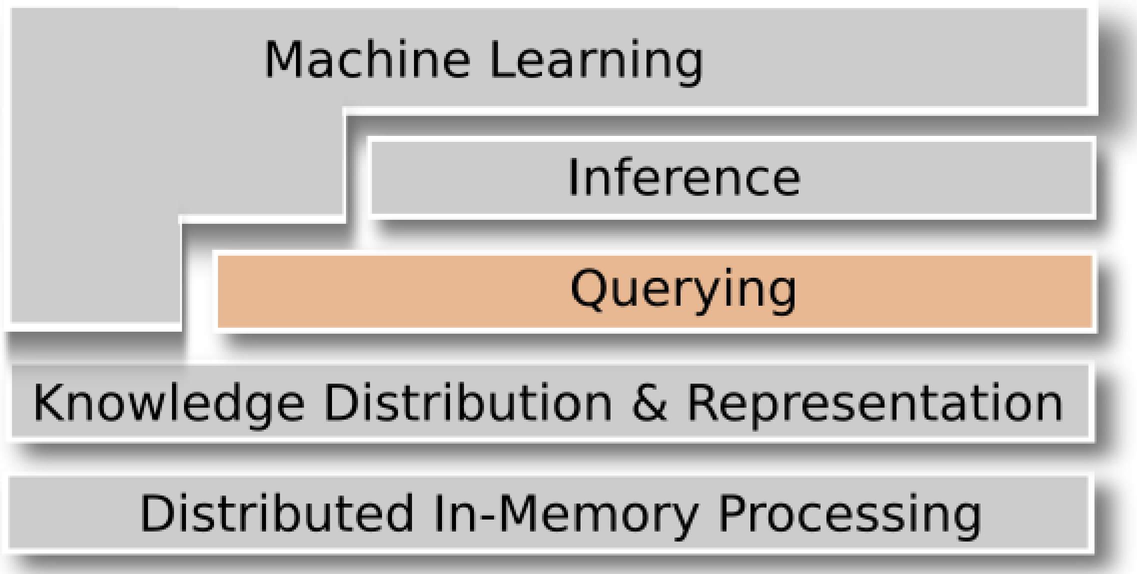
- T_{ijk} is 1 if triple (i-th entity, k-th predicate, j-th entity) exists and 0 otherwise





Learning, Applying, Multiplying Big Data Analytics

SANSA: Query Layer



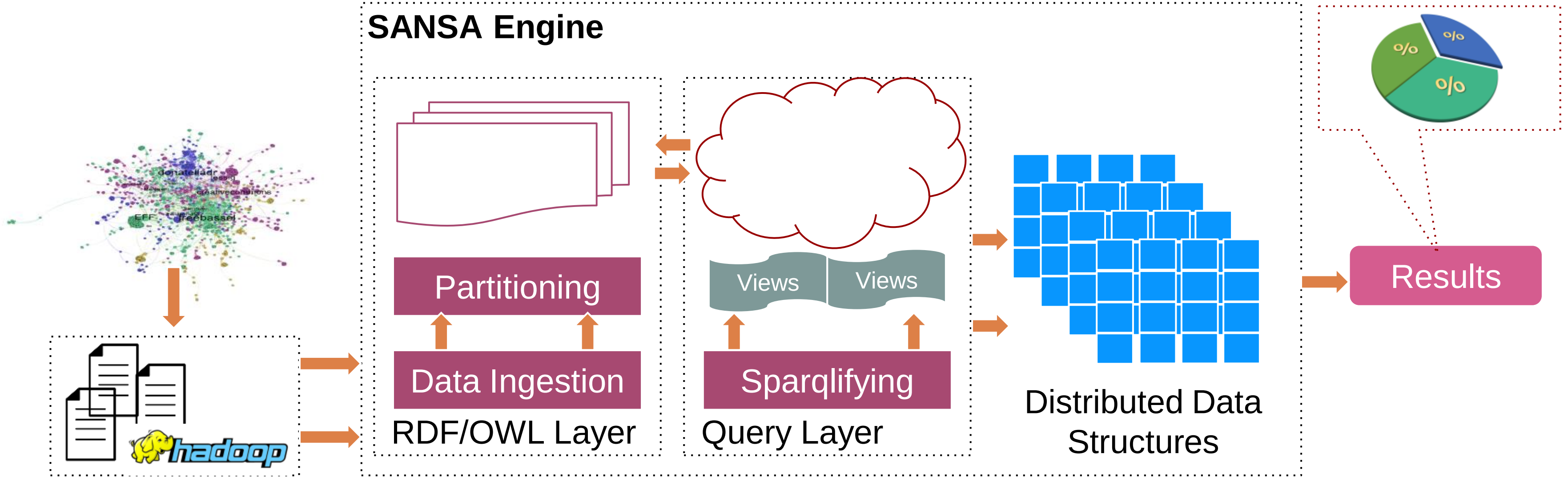
SANSA: Query Layer

- To make generic queries efficient and fast using:
  - Intelligent indexing
  - Splitting strategies
  - Distributed Storage
- SPARQL query engine evaluation
  - (SPARQL-to-SQL approaches, Virtual Views)
- Provision of W3C SPARQL compliant endpoint

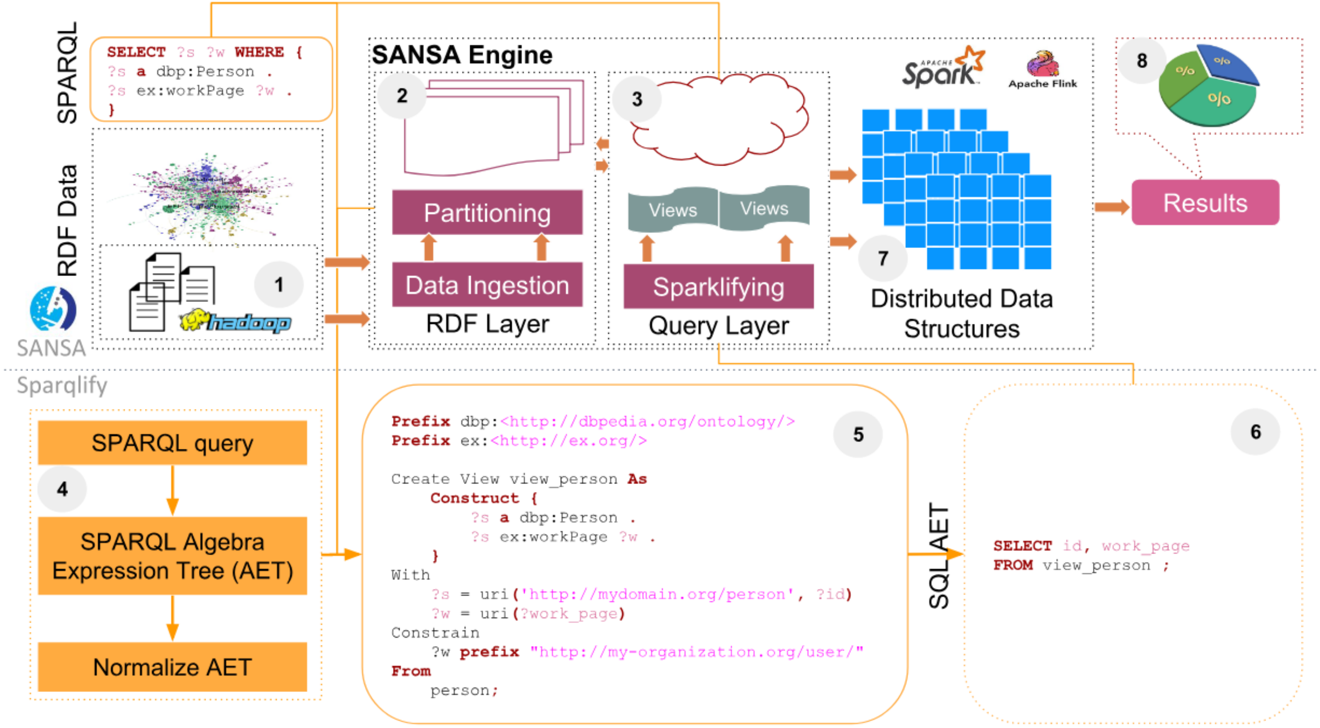
```
val triples = spark.rdf(Lang.NTRIPLES)(input)

val sparqlQuery = "SELECT * WHERE {?s ?p ?o} LIMIT 10"
val result = triples.sparql(sparqlQuery)
```

Querying via SPARQL & Partitioning



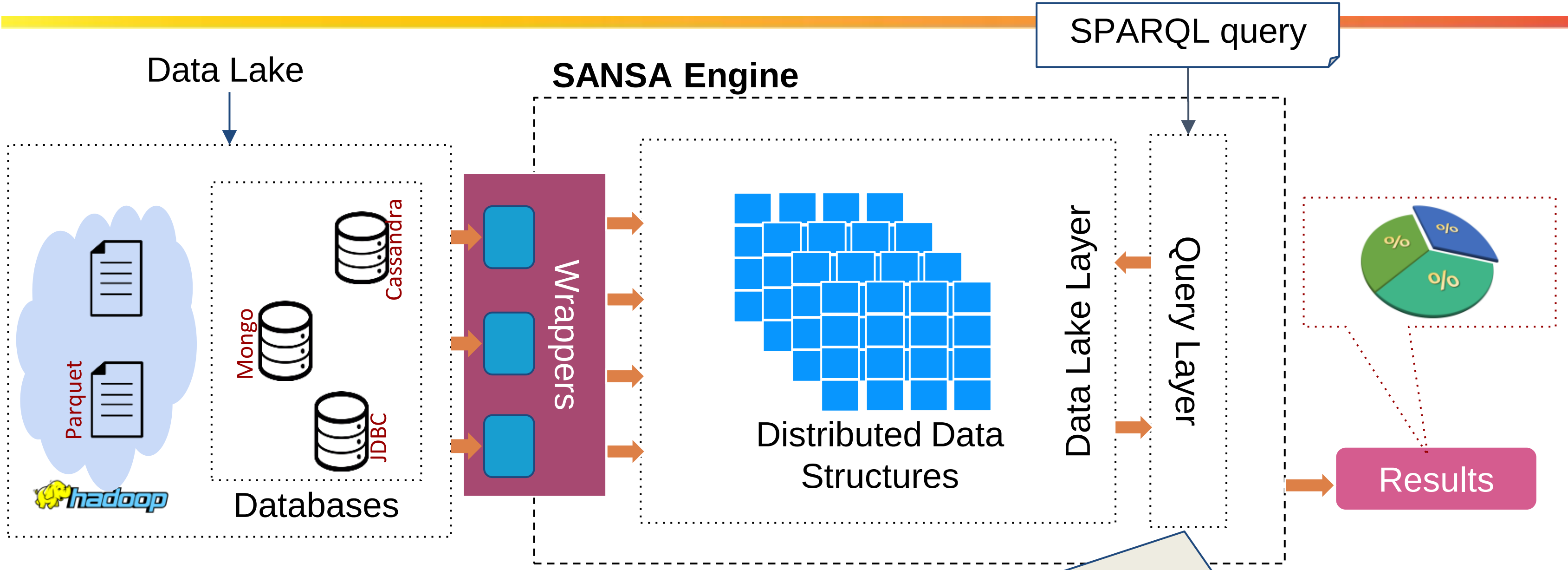
Querying via SPARQL & Partitioning



SANSA-DataLake

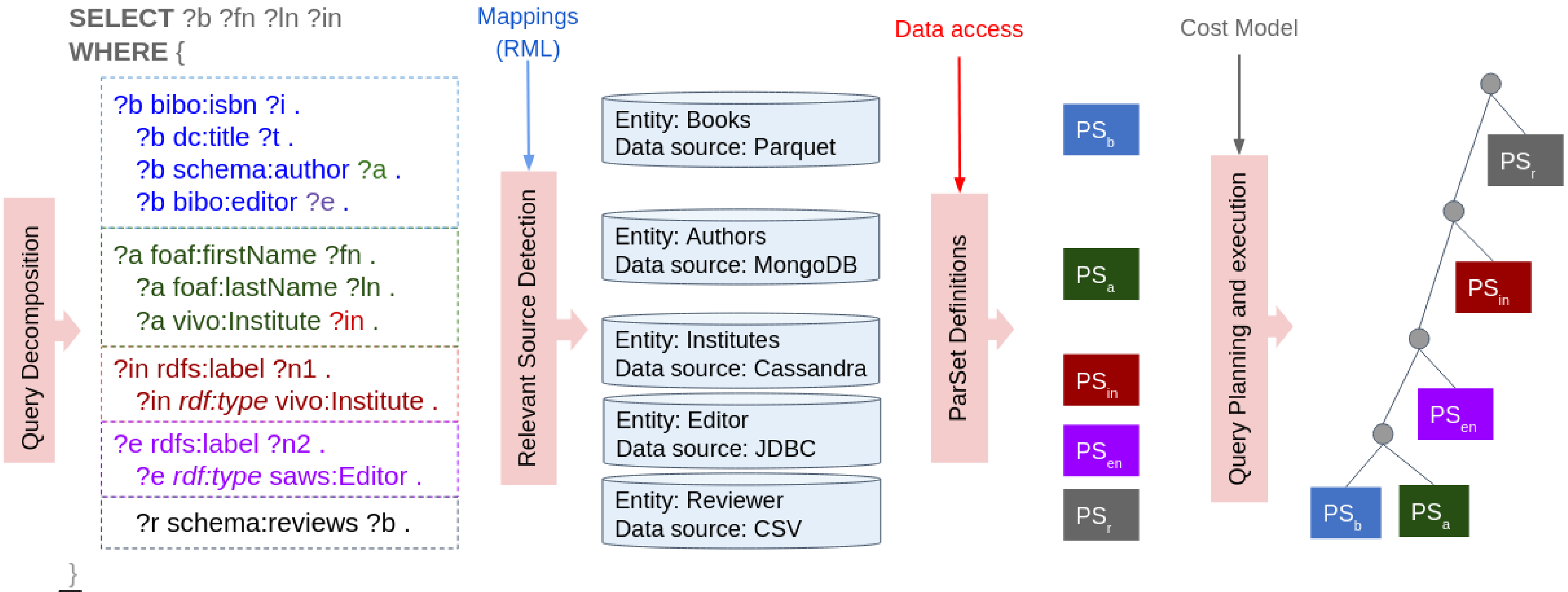
- A solution for the virtual Big Data integration: **Semantic Data Lake**
  - Directly query original data without prior transformation/loading
- Scalable cross-source query execution (join)
- Extensible (programmatically)
  - Do not reinvent the wheel: use existing engine connectors (wrappers)

Querying via Semantic Data Lake

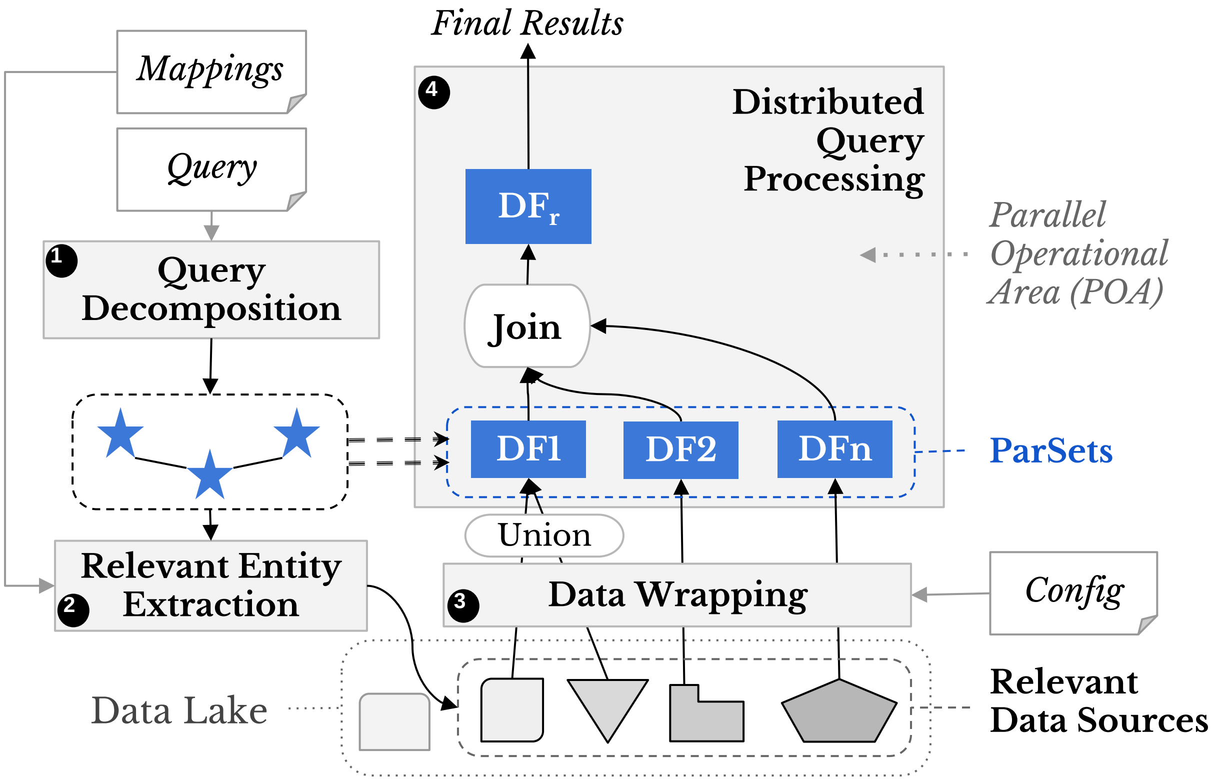



```
val result = spark.sparqlDL(query, mappings, config)
```

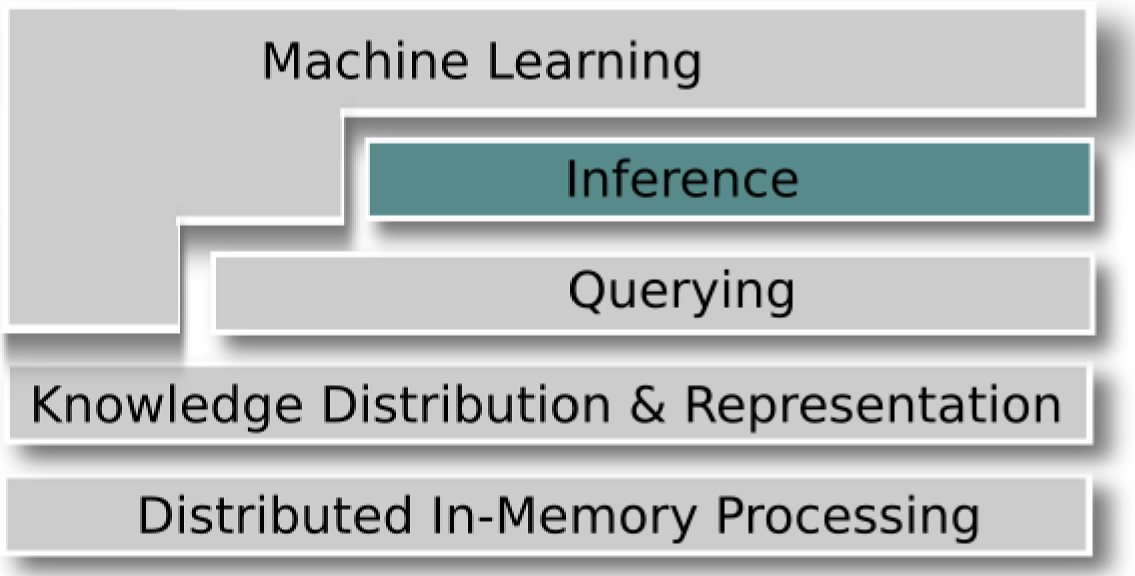
SANSA-DataLake



SANSA-DataLake



SANSA: Inference Layer



SANSA: Inference Layer

- The volume of semantic data growing rapidly
- Implicit information needs to be derived by reasoning
  - Reasoning

  \- the process of deducing implicit information from existing RDF data by using W3C Standards for Modelling: RDFS or OWL fragments

SANSA: Inference Layer

- **Reasoning** can be performed in two different strategies:
  - The forward chaining strategy derives and stores the derived RDF data back into original RDF data storage for late queries from applications (data-driven).
  - The backward chaining strategy derives implicit RDF data on the fly during query process (goal-directed).
- The forward chaining strategy has lower query response time and high load time.

SANSA: Inference Layer

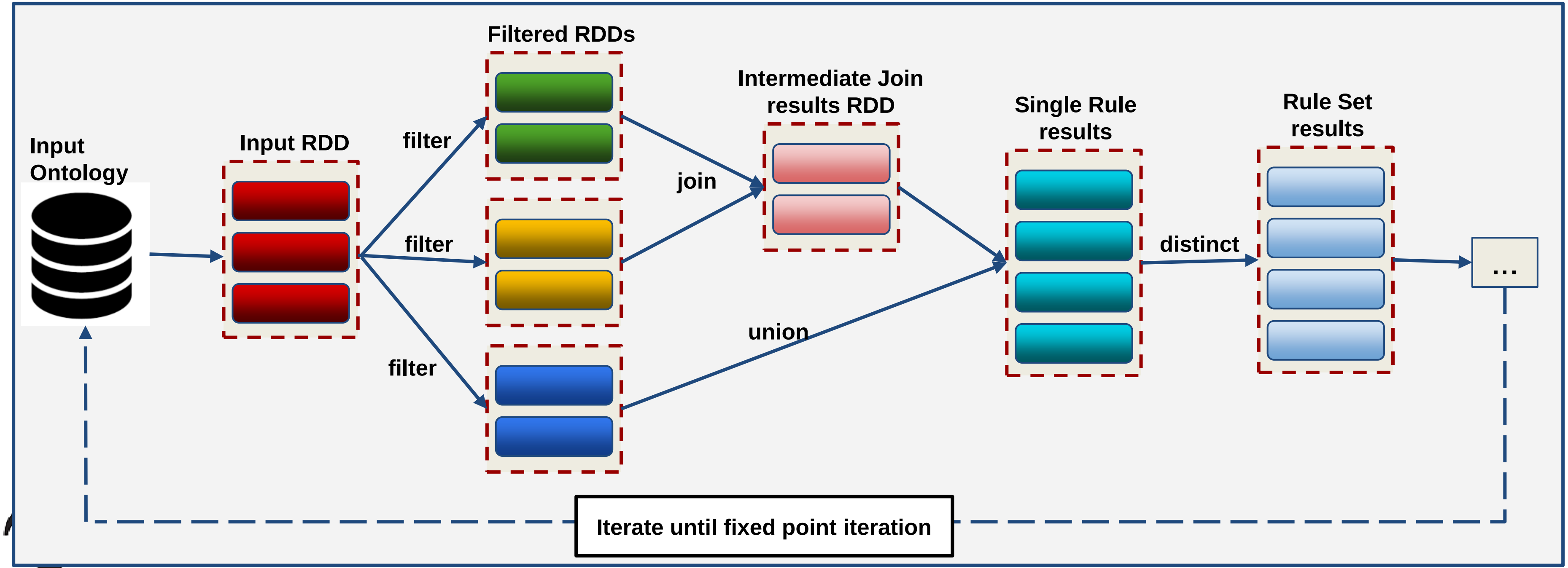
- Parallel in-memory inference via rule-based forward chaining
- Beyond state of the art: dynamically build a **rule dependency graph** for a rule set

  → Adjustable performance

  → Allows domain-specific customisation

SANSA: Inference Layer

Parallel RDFS Reasoning Algorithm based on Spark



SANSA: Inference Layer

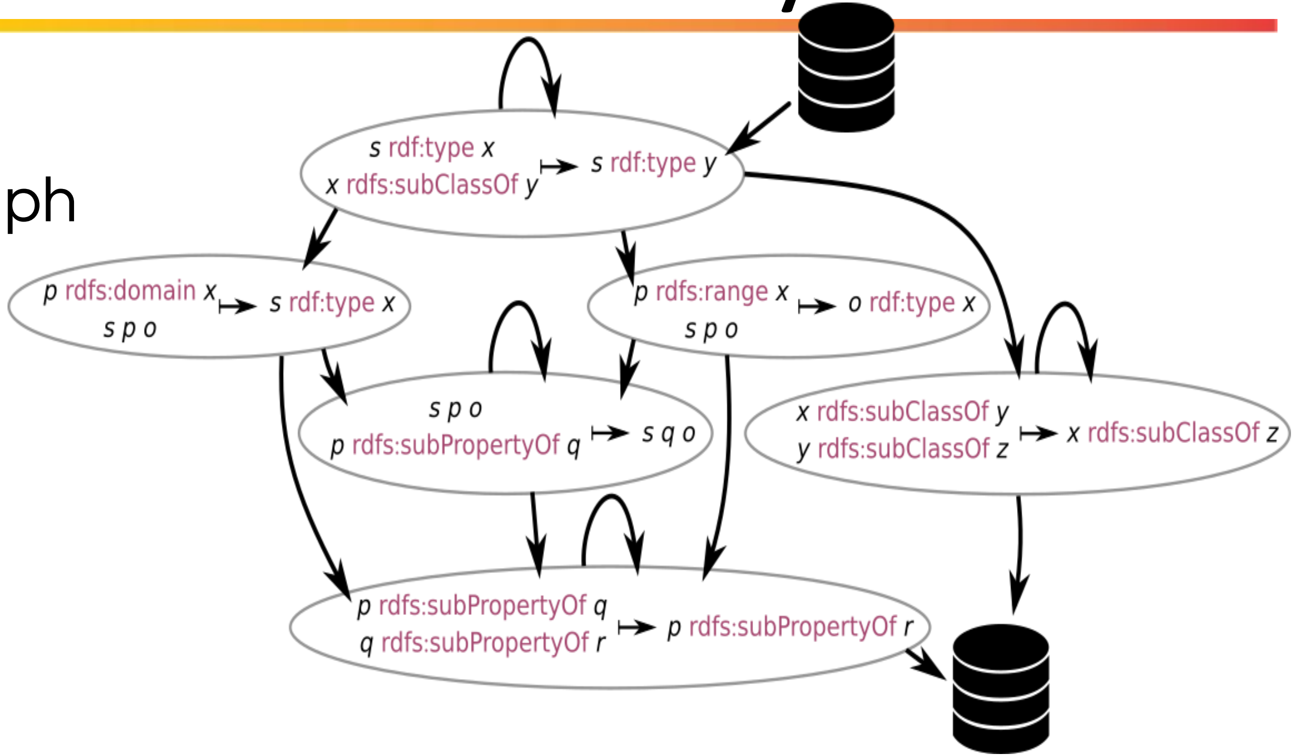
Some RDFS inference rules

- (X R Y), (R subPropertyOf Q) ➞ (X Q Y)
- (X R Y), (R domain C) ➞ (X type C)
- (X type C), (C subClassOf D) ➞ (X type D)

SANSA: Inference Layer

RDFS rule dependency graph

(simplified)



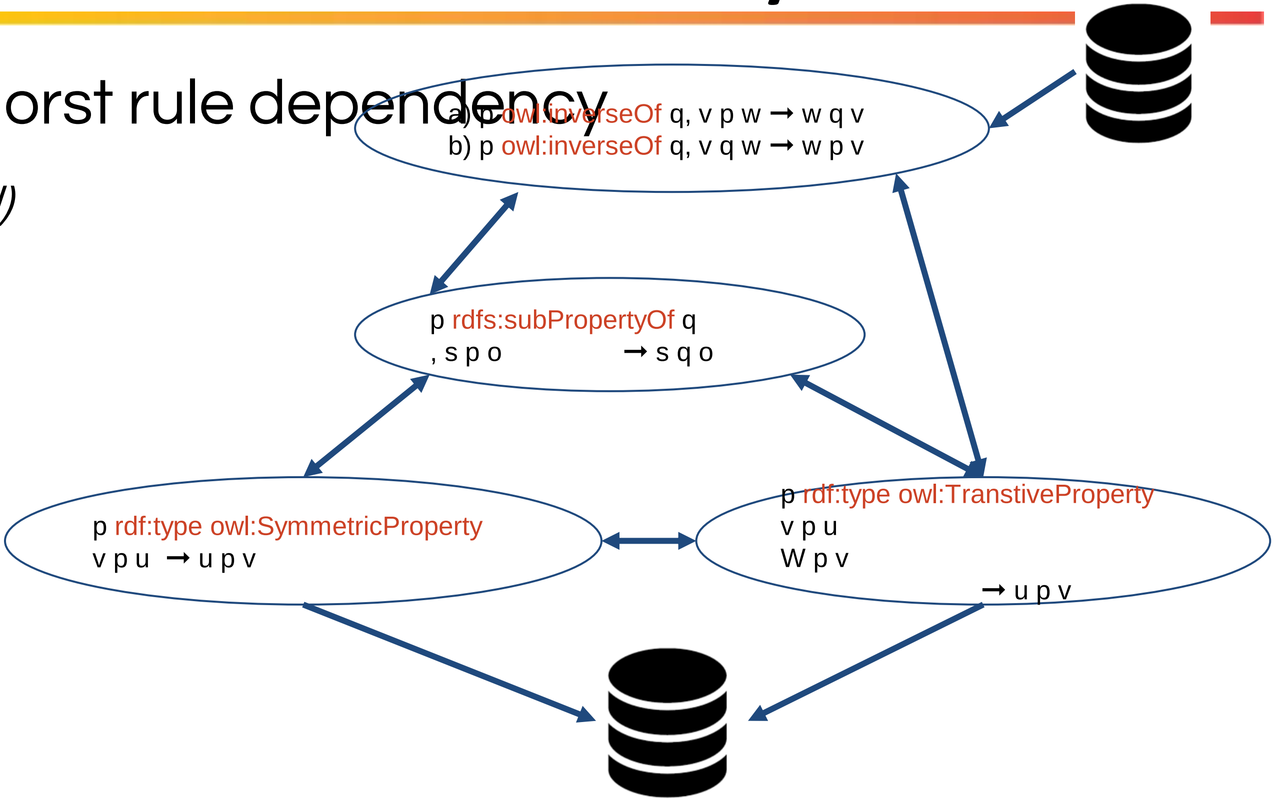
SANSA: Inference Layer

Some OWL Horst Rules:

- p owl:inverseOf q, v p w → w q v
- p owl:inverseOf q, v q w → w p v

SANSA: Inference Layer

- Part of OWL Horst rule dependency graph *(simplified)*



LEARNING, APPLYING, MULTIPLYING BIG DATA ANALYTICS

SANSA: Inference Layer

- SANSA-Inference Layer support RDFs and OWL-Horst reasoning in Triples and OWLAxioms

- Triple based forward chaining:

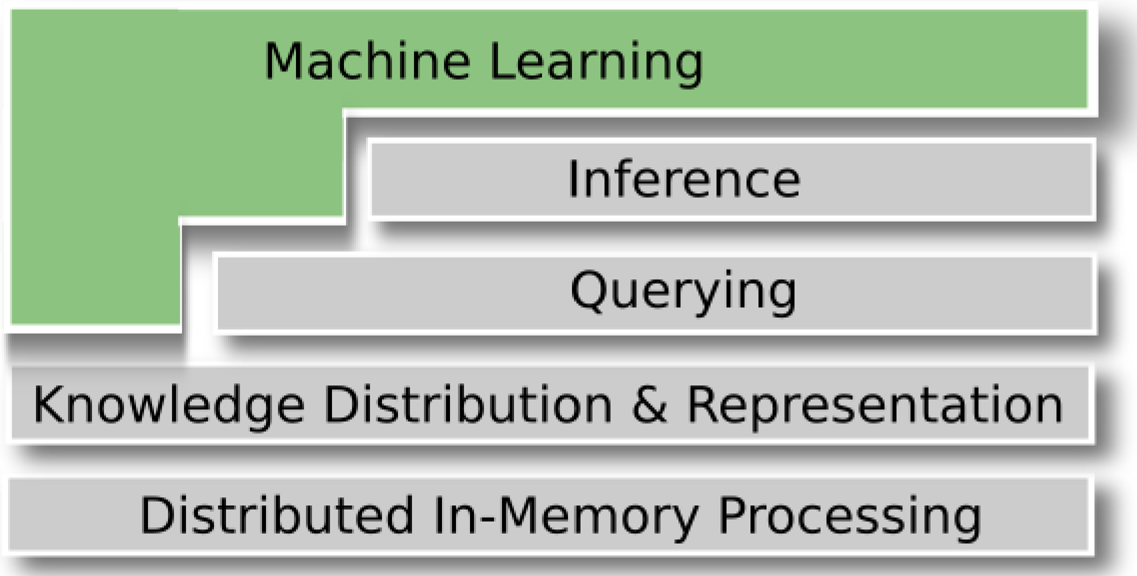
```
// load triples from disk
val graph = RDFGraphLoader.loadFromDisk(spark, input, parallelism)
val reasoner = new ForwardRuleReasonerOWLHorst(spark.sparkContext)
// compute inferred graph
val inferredGraph = reasoner.apply(graph)
```

SANSA: Inference Layer

- Axiom based forward chaining:

```
// load axioms from disk
var owlAxioms = spark.owl(Syntax.FUNCTIONAL)(input)
// create reasoner and compute inferred graph
val inferredGraph = profile match {
      case RDFS => new ForwardRuleReasonerRDFS(spark.sparkContext,
parallelism)(owlAxioms)
      case OWL_HORST => new
ForwardRuleReasonerOWLHorst(spark.sparkContext, parallelism)(owlAxioms)
      case _ => throw new RuntimeException("Invalid profile: '" +
profile + "'")
}
```

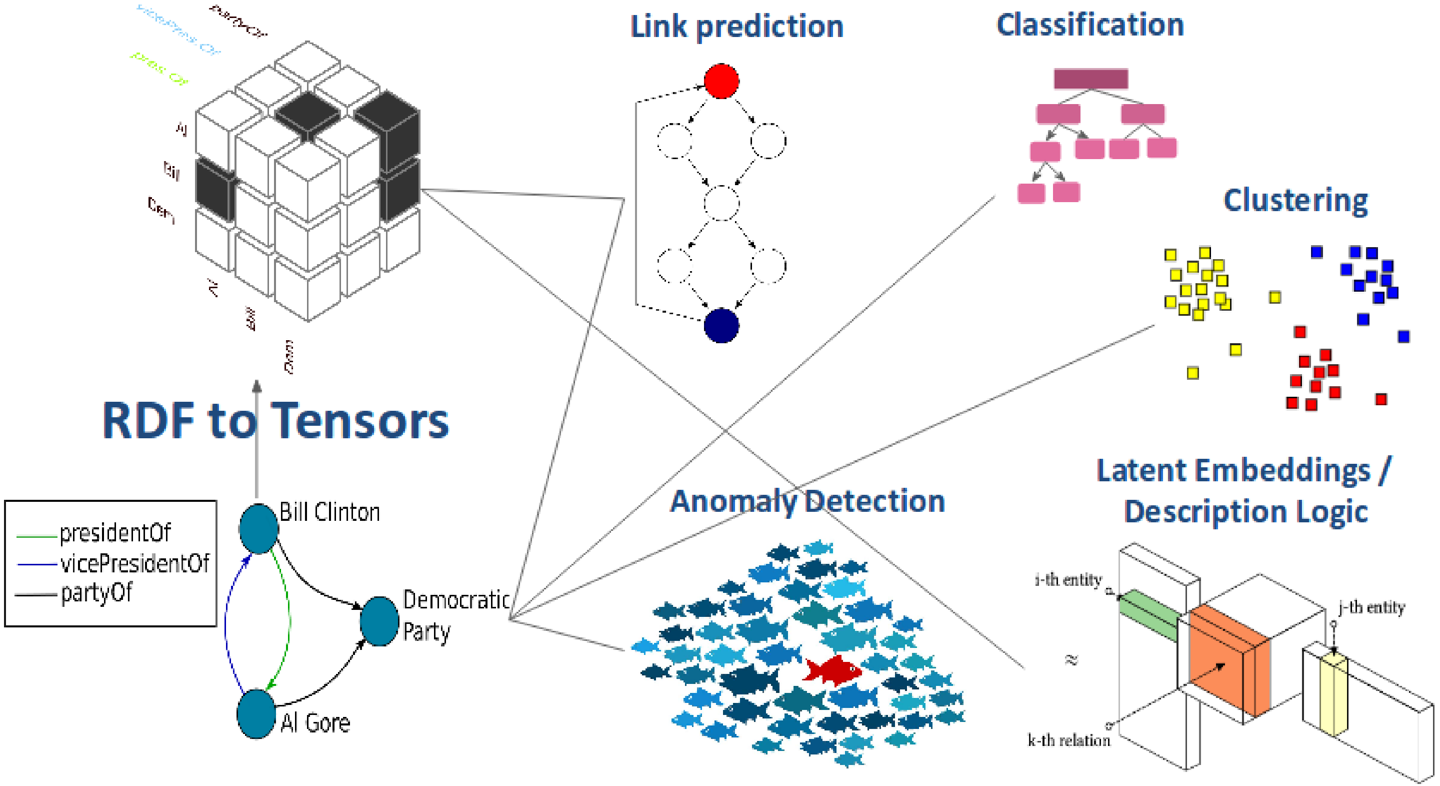
SANSA: ML Layer



SANSA: ML Layer

- Distributed Machine Learning (ML) algorithms that work on RDF data and make use of its structure / semantics
- Algorithms:
  - Graph Clustering
    - Power Iteration,
    - BorderFlow,
    - Link based
    - Modularity based clustering
  - Association rule mining (AMIE+ = mining horn rules from RDF data using partial completeness assumption and type constraints)
  - Outlier detection
  - KG Kernels

Scope



LEARNING, APPLYING, MULTIPLYING BIG DATA ANALYTICS

RDF By Modularity Clustering

- A Hierarchical clustering method
- Starts with each vertex in its own community
- Iteratively join pairs of community choosing the one with greatest increase in the optimizing function Q
  - Optimization function identifies the significant community structure
- The cut off point maximal value of Q.
- Scales as the square of the network size

```
RDFByModularityClusteringAlg(spark.sparkContext,
numIterations, input, output)
```

```
spark.stop
```

Power Iteration Clustering

- Simple and fast version of spectral clustering technique
- Efficient and scalable in terms of time O(n) and space
- Applying PowerIteration to the row normalized affinity matrix
- Partitioning clustering algorithm
  - Outputs one-level clustering solution

```
val lang = Lang.NTRIPLES
    val triples = spark.rdf(lang)(input)
    val graph = triples.asStringGraph()
    val cluster = RDFGraphPowerIterationClustering(spark, graph, output, k, maxIterations)
    cluster.saveAsTextFile(output)
```

BorderFlow Clustering

local graph clustering algorithm

Designing for directed and undirected weighted graphs

Clusters in BorderFlow:

Maximal intra-cluster density

Minimal outer-cluster density

```
val lang = Lang.NTRIPLES
val triples = spark.rdf(lang)(input)
val graph = triples.asGraph()

val borderflow = algName match {
             case "borderflow" => BorderFlow(spark, graph, output, outputevlsoft, outputevlhard)
             case "firsthardening" => FirstHardeninginBorderFlow(spark, graph, output, outputevlhard)
             case _ =>
             throw new RuntimeException("'" + algName + "' - Not supported, yet.")
             }
```

Link Based Clustering

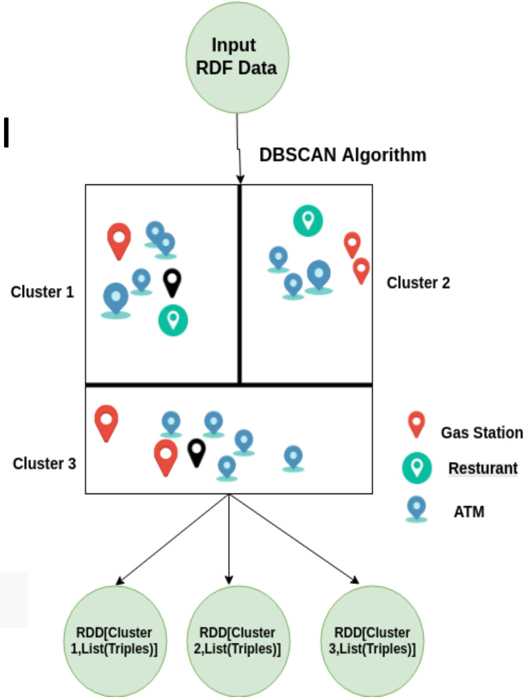
- Hierarchical link clustering method
- Bottom up approach of hierarchical called the "agglomerative"
- Clusters are created recursively
- The similarity S between links can be given by e.g. Jaccard similarity

```
1.    val lang = Lang.NTRIPLES
2.    val triples = spark.rdf(lang)(input)
3.    val graph = triples.asStringGraph()

4.    AlgSilviaClustering(spark, graph, output, outputeval)
```

DBSCAN Clustering

- Clustering of POIs on the basis of spatially closed coordinates.
- POIs located in same geolocation are clustei of their categories.




```
1.    val lang = Lang.NTRIPLES
2.    val triples = spark.rdf(lang)(input)
3.    val graph = triples.asStringGraph()
4.    AlgSilviaClustering(spark, graph, output,
outputeval)
```

Numerical Outliers Detection

- Detecting numerical outliers in large RDF dataset.
- Spark minHashLSH used to create the cohort of similar class.
- A scalable approach to find the outliers in a massive dataset.
- Numerical outliers detected in the data are useful for improving the quality of RDF Data

RDF Graph Kernels

Given an RDF graph constructing a tree for each instance and counting the number of paths in it.

Literals in RDF can only occur as objects in triples and therefore have no out-going edges in the RDF graph.

Apply Term Frequency-Inverse Document Frequency (TF-IDF) for vectors.

```
1. val rdfFastGraphKernel = RDFFastGraphKernel(spark, triples,
   "http://swrc.ontoware.org/ontology#affiliation")
2. val data = rdfFastGraphKernel.getMLLibLabeledPoints
3. RDFFastTreeGraphKernelUtil.predictLogisticRegressionMLLIB(data,
   4, iteration)
```

Rule Mining

- Association Rule Mining under Incomplete Evidence (AMIE)
- Atoms : are facts with the subject and object position substituted with variables e.g. (isChildOf(?a,?b))
- Rules : made up of atoms having a head (one atom) and the body (multiple atoms)
- The body $\{B_1, \ldots, B_n\}$ predicts the head $r(x,y)$
- A Rule can be written as

$$B_1 \wedge B_2 \wedge \ldots \wedge B_n \Rightarrow r(x, y)$$

- OR $\vec{B} \Rightarrow r(x, y)$

LEARNING, APPLYING, MULTIPLYING BIG DATA ANALYTICS

Interactive SANSA in your Browser

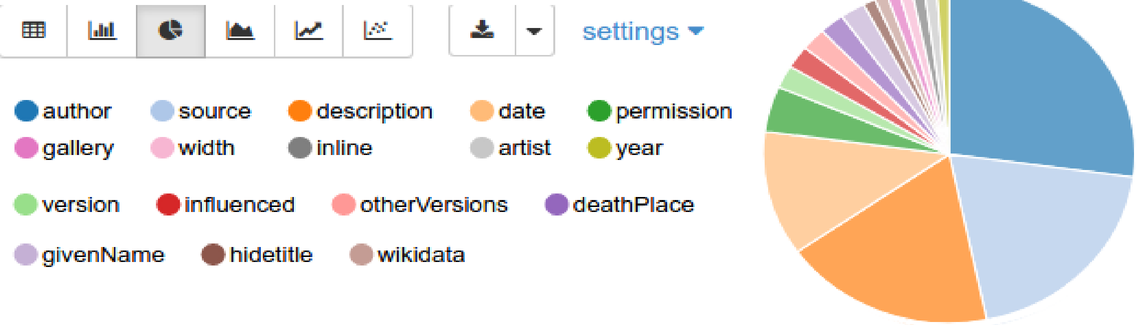
```
val input = "hdfs://namenode:8020/data/rdf.nt"
val triplesRDD = NTripleReader.load(spark, JavaURI.create(input))

val propertyDist = PropertyUsage(triplesRDD, spark).PostProc()
                                .map(f => f._1.getLocalName+ "\t" + f._2)

println("%table Property Distribution\tFrequency\n " + propertyDist.mkString("\n"))
```

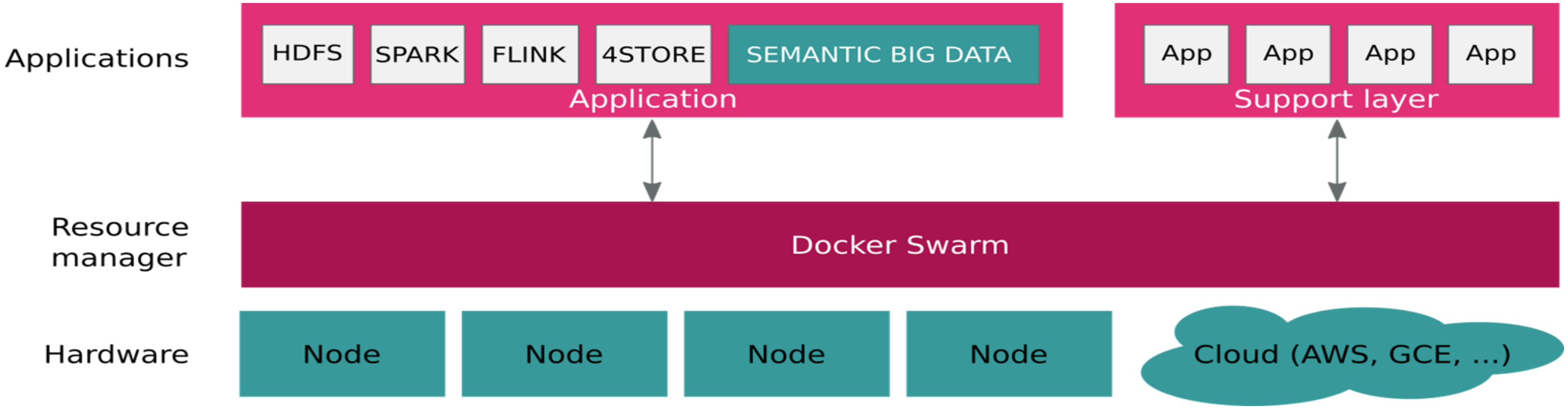
FINISHED

```
input: String = hdfs://namenode:8020/data/rdf.nt
triplesRDD: org.apache.spark.rdd.RDD[org.apache.jena.graph.Triple] = MapPartitionsRDD[27] at map at NTripleReader.scala:39
propertyDist: Array[String] = Array(author    25, source    18, description 17, date    11, permission 4, version  2, influenced  2,
2, deathPlace    2, givenName    2, hidetitle  1, wikidata  1, gallery      1, width  1, inline      1, artist   1, year  1)
```



BDE & General Integration

- SANSA = Scala / Maven Repositories based on Spark / Flink
- Easy to include both in BDE platform and any Spark / Flink environment



SANSA Planning and Pulse

- **SANSA 0.6 in June 2018, releases every 6 months**
- Apache Open Source License
- Project activity:
  - Contributors (*at least one commit*): **17**
  - Commits per day: **5.9 -** Commits previous year: **2175**
  - Github stars (all repos): **187**

Conclusions and Next steps

- A generic stack for (big) Linked Data
  - Build on top of a state-of-the-art distributed frameworks (Spark, Flink)
- Out-of-the-box framework for scalable and distributed semantic data analysis combining semantic web and distributed machine learning for (1) querying, (2) inference and (3) analytics of RDF datasets.
- Next steps
  - Support for SPARQL 1.1 and other partitioning strategies (Query Layer)
  - Backward chaining and better evaluation (Inference Layer)
  - More algorithms and definition of ML pipelines (ML Layer)

LEARNING, APPLYING, MULTIPLYING BIG DATA ANALYTICS

Associated Publications

1. **Distributed Semantic Analytics using the SANSA Stack** *by* Jens Lehmann, Gezim Sejdiu, Lorenz Bühmann, Patrick Westphal, Claus Stadler, Ivan Ermilov, Simon Bin, Muhammad Saleem, Axel-Cyrille Ngonga Ngomo and Hajira Jabeen in Proceedings of 16th International Semantic Web Conference – Resources Track (ISWC'2017), 2017 [BibTex].
2. **The Tale of Sansa Spark** *by* Ivan Ermilov, Jens Lehmann, Gezim Sejdiu, Lorenz Bühmann, Patrick Westphal, Claus Stadler, Simon Bin, Nilesh Chakraborty, Henning Petzka, Muhammad Saleem, Axel-Cyrille Ngomo Ngonga, and Hajira Jabeen *in* Proceedings of 16th International Semantic Web Conference, Poster & Demos, 2017 [BibTex].
3. **DistLODStats: Distributed Computation of RDF Dataset Statistics** *by* Gezim Sejdiu, Ivan Ermilov, Jens Lehmann, and Mohamed Nadjib-Mami *in* Proceedings of 17th International Semantic Web Conference, 2018. [BibTex]
4. **STATisfy Me: What are my Stats?** *by* Gezim Sejdiu; Ivan Ermilov; Jens Lehmann; and Mohamed-Nadjib Mami. In *Proceedings of 17th International Semantic Web Conference, Poster & Demos*, 2018.
5. **Profiting from Kitties on Ethereum: Leveraging Blockchain RDF with SANSA** *by* Damien Graux; Gezim Sejdiu; Hajira Jabeen; Jens Lehmann; Danning Sui; Dominik Muhs; and Johannes Pfeffer. In *14th International Conference on Semantic Systems, Poster & Demos*, 2018.
6. **SPIRIT: A Semantic Transparency and Compliance Stack** *by* Patrick Westphal, Javier Fernández, Sabrina Kirrane and Jens Lehmann. In *14th International Conference on Semantic Systems, Poster & Demos*, 2018.
7. **Divided we stand out! Forging Cohorts fOr Numeric Outlier Detection in large scale knowledge graphs (CONOD)** *by* Hajira Jabeen; Rajjat Dadwal; Gezim Sejdiu; and Jens Lehmann. In *21st International Conference on Knowledge Engineering and Knowledge Management (EKAW'2018)*, 2018.
8. **Clustering Pipelines of large RDF POI Data** *by* Rajjat Dadwal; Damien Graux; Gezim Sejdiu; Hajira Jabeen; and Jens Lehmann. In *ESWC 2019 (Poster Track)*.



Prof. Jens Lehmann Dr. Damien Graux Dr. Hajira Jabeen

THANK YOU !

LEARNING, APPLYING, MULTIPLYING BIG DATA ANALYTICS

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under grant agreement No 809965.