Building Knowledge Graph in Spark without SPARQL

Alex Romanova[1]

[1] Melenar, LLC, McLean, VA, US, 22101, sparkling.dataocean@gmail.com

Abstract. Knowledge graphs, powerful assets for enhancing search and various data integration, are being essential in both academia and industry. In this paper we will demonstrate that knowledge graph abilities are much wider than search and data integration. We will do it in a twofold manner: 1) we will show how to build knowledge graph in Spark instead of using SPARQL language and how to explore data in DataFrames and GraphFrames; and 2) we will reveal Spark knowledge graph as a bridge between logical thinking and graph thinking for data mining.

Keywords: Knowledge Graph, Spark, Scala, DataFrames, GraphFrames.

1. Introduction

On his keynote presentation on Semantics 2017 conference in Amsterdam, Aaron Bradley declared that "Semantic Web has died but in 2012 it was reincarnated by Google Knowledge Graph" [1]. Since that time knowledge graph was adapted by many companies as a powerful way to integrate and search various data. In this paper we will demonstrate some examples showing that knowledge graph abilities are much wider than search and data integration.

Just because Semantic Web was reborn to knowledge graph, does not mean that knowledge graph methodology is limited to Semantic Web methodology. In particularly, knowledge graphs with methods based on SPARQL language have many limitations [2] such as not supported language negation or predicate propertied. In this paper we will show how to build a knowledge graph in Spark Scala and how to explore data using Spark DataFrames and Spark GraphFrames.

Why Spark? Spark is a powerful open source analytics engine with libraries for SQL (DataFrames), machine learning, graphs (GraphFrames) that can be used together for interactive data exploration and supports wide array of data formats and storage systems [3], [4]. Until recently there were no processing frameworks that were able to solve several very different analytical problems like ETL, statistics and graphs in one place. Databricks supports running Spark code on Amazon Cloud via free Databricks Community [5].

DataFrames is a distributed collection of data organized into named columns [6] conceptually the same as table in a relational database. It can be constructed from a wide variety of sources and was created to leverage power of distributed processing frameworks to solve very complex problems.

Spark GraphFrames [7] is graph library build on top of DataFrames. It is a very powerful tool for performing distributed graph computations on big data that allows to combine graph in relational logic.

2. Artist Biography Knowledge Graph

2.1. Building Knowledge Graph in Spark

After knowledge graph was introduced by Google in 2012 it was adapted by many companies as Enterprise Knowledge Graph and is well known as a powerful engine for search and integration of various data [8].
Knowledge graph concept is much wider than search and data integration. We will show how knowledge graph can be used for data exploration to get a deeper view of data.
To illustrate how knowledge graphs can be built and explored in Spark, as a data domain we will use modern art movement data. First, we will create Artist Biography knowledge graph from a Kaggle dataset 'Museum of Modern Art Collection' [9]. Second, we will find connections between artists based on data from MoMA exhibition: 'Inventing Abstraction 1910-1925' [10]. And third, we will get names of key artists from data about timeline of the 'Modern Art Movements' [11].
From Kaggle dataset we extract biography data (Artist, ArtistBio, Nationality, BeginDate, EndDate, Gender) for several artists:

```
Claude Monet,"(French, 1840–1926)",(French),(1840),(1926),(Male)
Egon Schiele,"(Austrian, 1890–1918)",(Austrian),(1890),(1918),(Male)
Franz Marc,"(German, 1880–1916)",(German),(1880),(1916),(Male)
Georges Braque,"(French, 1882–1963)",(French),(1882),(1963),(Male)
Henri Matisse,"(French, 1869–1954)",(French),(1869),(1954),(Male)
Jackson Pollock,"(American, 1912–1956)",(American),(1912),(1956),(Male)
Joan Miró,"(Spanish, 1893–1983)",(Spanish),(1893),(1983),(Male)
Kazimir Malevich,"(Russian, born Ukraine. 1878–1935)",(Russian),(1878),
(1935),(Male)
Marc Chagall,"(French, born Belarus. 1887–1985)",(French),(1887),
(1985),(Male)
Max Beckmann,"(German, 1884–1950)",(German),(1884),(1950),(Male)
Natalia Goncharova,"(Russian, 1881–1962)”,(Russian),(1881),(1962),
(Female)
Oskar Kokoschka,"(Austrian, 1886–1980)",(Austrian),(1886),(1980),(Male)
Pablo Picasso,"(Spanish, 1881–1973)",(Spanish),(1881),(1973),(Male)
Paul Cézanne,"(French, 1839–1906)",(French),(1839),(1906),(Male)
Paul Gauguin,"(French, 1848–1903)",(French),(1848),(1903),(Male)
Paul Klee,"(German, born Switzerland. 1879–1940)",(German),(1879),
(1940),(Male)
Paul Signac,"(French, 1863–1935)",(French),(1863),(1935),(Male)
Piet Mondrian,"(Dutch, 1872–1944)",(Dutch),(1872),(1944),(Male)
Vasily Kandinsky,"(French, born Russia. 1866–1944)",(French),(1866),
(1944),(Male)
Vincent van Gogh,"(Dutch, 1853–1890)",(Dutch),(1853),(1890),(Male)
```

To build Spark knowledge graph on Artist - Nationality relationships, we will create nodes and edges by loosely following the RDF standards, i.e. (subject, object, predicate) form:

```
val graphNodesNationality=aboutArtist.select("Artist").
  union(aboutArtist.select("Nationality")).distinct.toDF("id")
val graphEdgesNationality=aboutArtist.select("Artist","Nationality").
```

```
  toDF("src","dst").withColumn("edgeId",lit("Nationality")).distinct
val graphNationality =
  GraphFrame(graphNodesNationality, graphEdgesNationality)
```

2.2. Knowledge Enrichment

Based on data, we can see that some artists changed their nationalities, for example, Marc Chagall, French, was born in Belarus in 1887. This is confusing because Belarus country was founded in 1990. Usually art museums display countries that artists' places of birth currently belong. To enrich knowledge graph code for artist's country of birth and born nationality, we change current country names to historical country names: changed "Russia","Ukraine", and "Belarus" to "Russian Empire". Then we map artist's country of birth to artist born nationality (bornNationality). Comparing artist's Nationality with bornNationality gives us a list of artists who changed their nationalities:

```
Artist, bornNationality, Nationality
Paul Klee, Swiss, German
Vasily Kandinsky, Russian, French
Marc Chagall, Russian, French
```

Spark code details can be found in our post [12].

2.3. Transform Tabular Data to Knowledge Graph

Another way to compare Spark knowledge graph method with SPARQL is building a knowledge graph from tabular data. In Spark it is very easy to automate this process. First, we will get column names:

```
val columnList=artistBio.columns
columnList: Array[String] = Array(Artist, BeginDate, EndDate,
Nationality, Gender, bornInCountry, bornNationality)
```

Next, for all columns we will create pair edges {'Artist', 'other columns'} and nodes {'Artist'}, {'other columns'}:

```
var graphEdges: DataFrame = Seq(("","","")).toDF("src","dst","edgeId")
var idx=0
for (column <- columnList) {
  graphNodes=graphNodes.union(artistBio.select(column))
  if (idx>0) {
    graphEdges=graphEdges.union(artistBio.
    select(artistBio.columns(0),column).
    toDF("src","dst").withColumn("edgeId",lit(column)))
  }
  idx=idx+1
}
```

Finally we will build a knowledge graph:

```
val graphNodesArtistBio=graphNodes.filter('id!="").distinct
val graphEdgesArtistBio=graphEdges.filter('src!="").distinct
val graphArtistBio =
  GraphFrame(graphNodesArtistBio, graphEdgesArtistBio)
```

Spark code details can be found in our post [12].

2.4. Graph Visualization

To visualize graph (Figure 1) we will translate graph edges to DOT language - graph description language for graph visualization [13]. For graph visualization we used Gephi tool [14].

Graph edges to DOT language code:

```
display(graphArtistBio.edges.filter('edgeId==="bornNationality").
  map(s=>("\""+s(0).toString +"\" -> \""+s(1).toString +"\""+" ;")))
"Natalia Goncharova" -> "Russian" ;
"Georges Braque" -> "French" ;
"Egon Schiele" -> "Austrian" ;
"Franz Marc" -> "German" ;
"Paul Klee" -> "Swiss" ;
"Marc Chagall" -> "Russian" ;
"Joan Miró" -> "Spanish" ;
"Vincent van Gogh" -> "Dutch" ;
"Kazimir Malevich" -> "Russian" ;
"Pablo Picasso" -> "Spanish" ;
"Oskar Kokoschka" -> "Austrian" ;
"Vasily Kandinsky" -> "Russian" ;
```

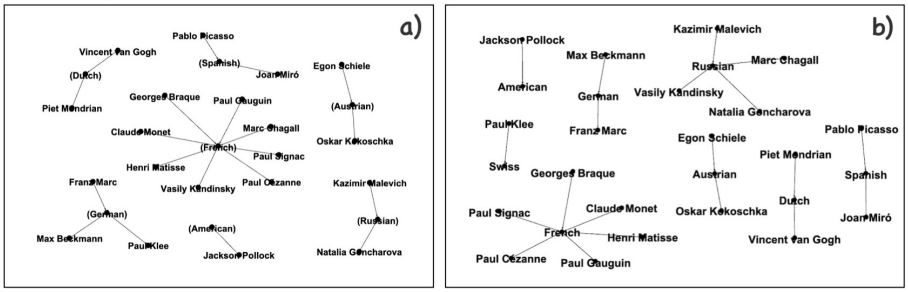

Fig. 1. Subgraphs on Artist Bio knowledge graph: a) Artists Nationalities from Kaggle dataset; b) Artists Born Nationalities.

2.5. Knowledge Graph Queries

Here are two examples of knowledge graph queries that are using Spark instead of 'traditional' SPARQL language. Let's say we need to find pairs of artists that were born in the Austria.

First method is using Spark DataFrames language by self-joining ArtistBio table:

```
val sameCountryBirthDF = artistBio.select("Artist","bornInCountry").
  join(artistBio.select("Artist","bornInCountry").
  toDF("Artist2","bornInCountry2"),'bornInCountry==='bornInCountry2).
  filter('Artist<'Artist2).
  select("bornInCountry","Artist","Artist2").distinct
```

```
Austria, Egon Schiele, Oskar Kokoschka
```

Second method is using Spark GraphFrames motif 'find' function [15]. This method is conceptually similar to {subject - predicate -> object} and it is better understandable and more elegant than self-joining of tabular data:

```
val sameCountryBirthGF=graphArtist.
  find("(a) - [ac] -> (c); (b) - [bc] -> (c)").
  filter($"ac.edgeId"===$"bc.edgeId" && $"ac.edgeId"==="bornInCountry").
  filter($"a.id"<$"b.id").select("c.id","a.id","b.id").
  toDF("bornInCountry","Artist1","Artist2").distinct
```

```
Austria, Egon Schiele, Oskar Kokoschka
```

3. Modern Art Movement Knowledge Graph

3.1 Semi-structured Data

To show how to use knowledge graph to integrate different data, first we will build a knowledge graph of modern art key artists. From semi-structured dataset about timeline of the Modern Art Movements [11] we will get names of key artists of modern art movements. From this list we will get a subset of artists from our artist biography knowledge graph. In our post [16] you can find data processing code in details.

```
1872 – 1892, Impressionism, Claude Monet
Early 1880s - 1914, Post-Impressionism, Paul Gauguin
Early 1880s - 1914, Post-Impressionism, Paul Signac
Early 1880s - 1914, Post-Impressionism, Paul Cézanne
1905 – 1910, Fauvism, Henri Matisse
1907 – 1922, Cubism, Pablo Picasso
1907 – 1922, Cubism, Georges Braque
1909 – late 1920s, Futurism, Natalia Goncharova
1913 – late 1920s, Suprematism, Kazimir Malevich
1917 – 1931, De Stijl, Piet Mondrian
1924 – 1966, Surrealism, Joan Miró
1943 – 1965, Abstract Expressionism, Jackson Pollock
```

Knowledge graph edges:

```
val modernArtEdges=
   modernArtData.select("keyArtist","artMovement").toDF("src","dst").
  withColumn("edgeId",lit("artMovement")).
  union(modernArtData.select("artMovement","time").toDF("src","dst").
  withColumn("edgeId",lit("time"))).distinct
```

Modern Art Movement knowledge graph:

```
val modernArtGraph=GraphFrame(modernArtEdges.select("src").
   union(modernArtEdges.select("dst")).distinct.toDF("id"),
  modernArtEdges.distinct)
```

3.2 Knowledge Graph Integration

There are two ways to integrate two knowledge graphs: combine edges of both graphs or overlap vertices of both graphs. To integrate Modern Art Movement knowledge graph with Artist Biography knowledge graph we will follow the second way.
From Artist Biography knowledge graph we will take only information about nationalities and countries where artists were born.

```
val modernArtBioEdges = graphArtistBio.edges.
  join(modernArtKeyArtists,'src==='keyArtist).drop("keyArtist").
  union(modernArtGraph.edges).
  filter(not('edgeId.isin("EndDate","BeginDate","Gender")))
```

Build a knowledge graph:

```
val modernArtBioGraph=GraphFrame(
  modernArtBioEdges.select("src").
  union(modernArtBioEdges.select("dst")).distinct.toDF("id"),
  modernArtBioEdges)
```

Integrated knowledge graph shows biographies of Modern Art Movements key artists (Figure 2).

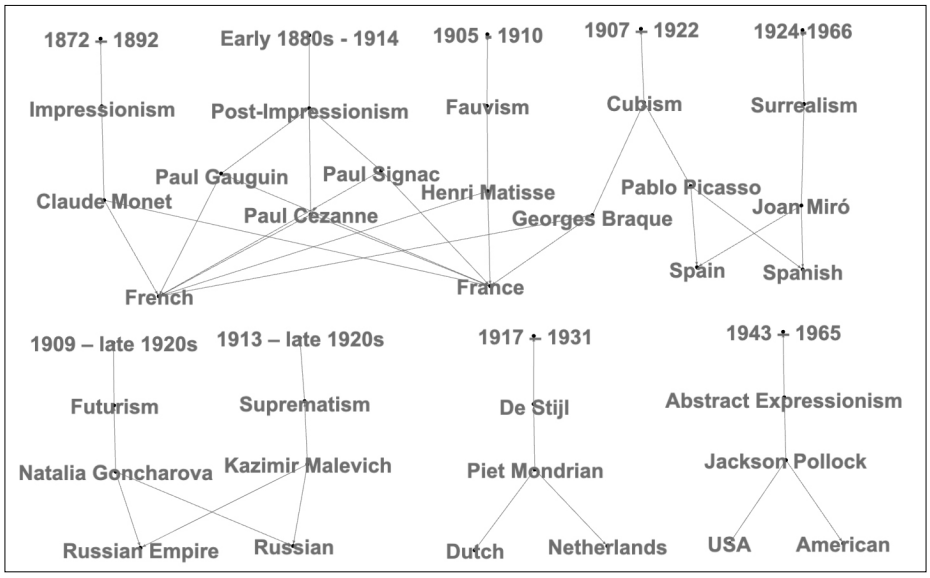

Fig. 2. Relationships between modern art movements.

This graph shows unknown connections between modern art movements: Impressionism, Post-Impressionism, Fauvism, Cubism and Surrealism were created by artists born in France or Spain. These art movements had no connections with Futurism and Suprematism that were created by artists born in Russian Empire.

3. Conclusion

In this paper we illustrated how Spark knowledge graph can build a bridge between logical thinking and graph thinking for data exploration. In data mining examples we demonstrated that Spark GraphFrames library provides a flexibility to switch between SQL functions and graph functions.
Exploring data about modern art artists we found unknown connections between artists and between modern art movements.

References

1. Aaron Bradley, Semantics 2017, https://2017.semantics.cc/aaron-bradley-eamonn-glass
2. The Limitations of SPARQL, http://horicky.blogspot.com/2010/08/limitations-of-sparql.html.
3. Apache Spark, https://databricks.com/spark/about
4. 'Spark: The Definitive Guide: Big Data Processing Made Simple', by Bill Chambers, Matei Zaharia
5. Databricks Community Edition, https://databricks.com/blog/2016/02/17/introducing-databricks-community-edition-apache-spark-for-all.html
6. Spark DataFrames, https://databricks.com/blog/2015/02/17/introducing-dataframes-in-spark-for-large-scale-data-science.html
7. Spark GraphFrames, https://databricks.com/blog/2016/03/03/introducing-graphframes.html
8. Industry-scale Knowledge Graphs: Lessons and Challenges, https://queue.acm.org/detail.cfm?id=3332266
9. Kaggle dataset 'Museum of Modern Art Collection', https://www.kaggle.com/momanyc/museum-collection.
10. MoMA exhibition: 'Inventing Abstraction 1910-1925', https://www.moma.org/interactives/exhibitions/2012/inventingabstraction/?page=artists
11. Timeline of the 'Modern Art Movements', https://drawpaintacademy.com/modern-art-movements/
12. "Knowledge Graph for Data Integration" post, http://sparklingdataocean.com/2020/02/02/knowledgeGraphIntegration/
13. Drawing graphs with dot, https://www.ocf.berkeley.edu/~eek/index.html/tiny_examples/thinktank/src/gv1.7c/doc/dotguide.pdf
14. Visual network analysis with Gephi, https://medium.com/@EthnographicMachines/visual-network-analysis-with-gephi-d6241127a336
15. Motifs Findings in GraphFrames, https://www.waitingforcode.com/apache-spark-graphframes/motifs-finding-graphframes/read
16. "Knowledge Graph for Data Mining" post, http://sparklingdataocean.com/2019/09/24/knowledgeGraphDataAnalysis/