Data Analytics using *KNIME* open source tool

Dörheit, Eric
TU Berlin
Berlin, Germany
eric.doerheit@campus.tu-berlin.de

Bode, Olga
TU Berlin
Berlin, Germany
olga.bode@mailbox.tu-berlin.de

Poljak, Dorothea
TU Berlin
Berlin, Germany
@mailbox.tu-berlin.de

ABSTRACT

We present our results of the evaluation of the open source tool *KNIME* which is used for data analytics and data mining. We choose anomaly detection as the subject to evaluate *KNIME* as many methods of data analytics such as clustering, classification, time series analysis and statistical techniques are applicable to anomaly detection [1]. As a data set for the analysis we use the data provided for the *DEBS Grand Challenge 2012* [2].

1. INTRODUCTION

For the evaluation of *KNIME* we first investigated the tool by following several white-papers provided by *KNIME*. We start with describing the functionalities and the usage of *KNIME*. Then we explain the data used for the evaluation. Subsequently we provide an overview on anomaly detection based on [1]. In Section 2 we evaluate *KNIME* through applying anomaly detection on the data set described in Section 1.2.

1.1 The Open Source Tool KNIME

In this section we give an overview on *KNIME* based on three white-papers provided by *KNIME*:

- Big Data, Smart Energy, and Predictive Analytics
- Anomaly Detection in Predictive Maintenance
- KNIME opens the Doors to Big Data

TODOS:

- General features of *KNIME* → what can you do with *KNIME* (ETL, Mining, Analysis, Visualization etc.)
- How to use *KNIME*? → Workflows, Nodes, ... (describe the usage of *KNIME* in general)
- Example workflow(s) based on the *KNIME* white-papers
- Overview on the nodes that are available and name, that there is an API to develop your own nodes
- Tiny summary (2 sentences) if possible

1.2 DEBS 2012 Grand Challenge

TODOS:

- Describe the challenge / the origin of the data
- Explain the data set

1.3 Anomaly Detection

Anomalies in data are patterns which do not conform to the expected behavior and anomaly detections deals with finding this patterns [1]. There are many techniques that can be applied to detect anomalies. Subsequently we describe classification based, nearest neighbor based and clustering based techniques as well as statistical anomaly detection techniques.

1.3.1 Based on Classification

1.3.2 Based on Nearest Neighbor

1.3.3 Based on Clustering

Clustering is a technique that assigns data instances to clusters. There are clustering algorithms which assign each data instance to a cluster and also algorithms which let data instances unassigned. In order to use a clustering as a means to detect anomalies, it is necessary to assume what an anomaly is in the context of clustering. There are three assumption usable for this purpose [1]:

1. Normal data instances belong to clusters while anomalies do not.
2. Normal data instances can be found closest to a cluster centroid while anomalies are far away from their closest cluster centroid.
3. Normal data instances lie in dense clusters while anomalies are in small or sparse clusters.

Assuming the first statement, algorithms such as DBSCAN, ROCK and SNN clustering are usable as they do not assign each data instance to a cluster. Based on the second assumption algorithms such as Self-Organized Maps (SOM), K-Means clustering as well as Expectation Maximization (EM) can be used due to the fact that they not only assign a data instance to a cluster but also compute its distance to its closest cluster centroid. Given the third assumption the use of various clustering algorithms is possible but it is needed to additionally compute the density/size of each

cluster and define thresholds below which the data instances of these clusters are anomalies.

1.3.4 Statistical Anomaly Detection Techniques

2. ANOMALY DETECTION WITH KNIME

In the following we describe how to detect anomal data instances in the *DEBS 2012 Grand Challenge* data set using *KNIME*. Therefore we start with explaining the handling of the large data set and proceed with the ETL process. Subsequently, we focus on the application of the techniques illustrated in Section 1.3.

2.1 Handling of large Data Sets

KNIME offers many possibilities to pull data out of different data sources as described in Section 1.1. Since *KNIME* stores data either in-memory or on the local filesystem, it is not possible to directly take advantage of a a distributed filesystem such as *HDFS*[1] within *KNIME*. Hence, we need a way to handle large files.

2.1.1 Split into smaller Files

The first solution is simply splitting the large file into smaller ones, e.g. with a command-line tool such as *split* or *awk*, and then making use of the loop and file list nodes in *KNIME* to iterate over the newly created files and execute the analytics on each of the files.

2.1.2 Use of Databases

The second option is to use a database to store the data set. Therefore, you can also split a large file into smaller ones, iterate over them and append a database table with the files data. An other way is not to use *KNIME* and to directly import the file into the database. In our case we use a *MySQL* database and the workflow shown in Figure **??**. To import data from a database into *KNIME* there are multiple nodes available to build SQL queries and execute them. We use a loop to import chunks of data so that we do not run out of memory as this decreases the performance of *KNIME*.

2.1.3 Big Data Extensions

An alternative for handling large files is to make use of the Big Data Extensions which *KNIME* offers. They enable to connect to a *HDFS*, where you can upload files to or read files from, and to *Hive* as well as *Impala*, which allows to execute SQL queries on an *Hadoop* system. These extensions make it possible to access data from these sources but do not accelerate *KNIME* in the favor of executing *KNIME* workflows in a cluster.

2.2 The ETL Process with KNIME

ETL describes the process of extracting, transforming and loading data. In the extractions phase data is extracted from various data sources. In our case we already have one homogeneous file. Then the transformation phase follows where the data is altered in order to be able to run queries and do analysis. Therefore, we split the timestamp into multiple columns and filter columns which remain constant. Afterwards we store the new tables into a *MySQL* database.

2.3 Anomaly Detection Techniques

[1]Reference HDFS paper

2.3.1 Classification

2.3.2 Clustering

In Section 1.3.3 we describe multiple approaches to detect anomalies with clustering. We apply these techniques on the data explained in Section 1.2.

2.3.3 Statistical Anomaly Detection Techniques

TODOS:

- How to input the big files into *KNIME*? → Split files and iterate over them, input into MySQL etc.
- Describe the ETL process with *KNIME* of the data
- Describe how to do anomaly detection with *KNIME*

3. RESULTS

TODOS:

- *KNIME* is not directly suitable for Big Data Processing
- Easy tool to do advanced data analytics without deep knowledge of underlying algorithms and math
- Enables users which are no data scientists or have a strong background in this field to do data analysis
- Good integration with various other tools (R, Weka) and adoptable to own needs with Java, Python, ... snippet nodes and the API to create own nodes
- Relatively slow

4. CONCLUSIONS

5. REFERENCES

[1] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

[2] Z. Jerzak, T. Heinze, M. Fehr, D. Gröber, R. Hartung, and N. Stojanovic. The debs 2012 grand challenge. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*, DEBS '12, pages 393–398, New York, NY, USA, 2012. ACM.