# A Novel method for Information Extraction and Visualization of the Learning Analytics and Knowledge Dataset

Nitin V Pujari
P E S Institute of Technology
Banashankari 3rd Stage
Bangalore, India
nitin.pujari@pes.edu

Bhargav H S
P E S Institute of Technology
Banashankari 3rd Stage
Bangalore, India
bhargavraohs@gmail.com

Gangadhar Akalwadi
P E S Institute of Technology
Banashankari 3rd Stage
Bangalore, India
gangadhar.ak55@gmail.com

Srinidhi R
P E S Institute of Technology
Banashankari 3rd Stage
Bangalore, India
srinnidhi@gmail.com

## ABSTRACT

Learning analytics in simple words is that principle which governs the analysis of acquiring knowledge. It is a relatively new field developed very recently due to increased research in the field of Analysis.

In the study of Learning Analytics the requirements of datasets is very important. The LAK Dataset provided by the Learning Analytics Summer Institute (LASI) is a very well defined and maintained dataset. The dataset provided by LASI is in nTriples format which is suitable for semantic web technologies. This particular format i.e. N-Triples is not supported by normal graph databases. Hence, our aim is to convert the data in N-Triples format to Cypher Text, a format very easy to learn and understand, and then uploading the dataset into a graph database management system called Neo4j which is freely available on multiple development platforms.

Apart from uploading the dataset into a Graph DBMS, we also aim at providing support to query the dataset using Cypher language either directly (using the Neo4j web view) or through a client and download whatever information that he requires in a tabular form as a portable document (PDF). This work carried out proposes a novel method for extracting information and providing visualization of the Learning Analytics and Knowledge dataset.

## General Terms

Learning Analytics

## Keywords

LAK-Dataset, Data Extraction, Visualization, Neo4j

## 1. INTRODUCTION

Learning Analytics as defined in the first international LAK-Conference, *is the measurement, collection, analysis and reporting of data about learners and their contexts, for purposes of understanding and optimizing learning and the environments in which it occurs.*[2] In the last few years there is a significant change in the field of research analytics. There is a huge demand and support for more and more research works related to analysis in different areas. One such area is the field of learning. The advantages of analyzing learning patterns helps teachers to predict the type of coaching the student requires and what are his interests.

The analysis of any subject requires the availability of large datasets. The Society for Learning Analytics Research (SoLAR) and Learning Analytics Summer Institute (LASI) ( along with CNR-ITD and the LinkedUp project) have provided us with a fairly good amount of data. This dataset is known as the Learning Analytics Knowledge dataset (or simply the LAK Dataset). The LAK dataset consists of details of research papers publised in the LAK (Learning Analytics Knowledge) and EDM (Educational Data Mining) conferences. It also provides metadata about the resource and is well maintained by SoLAR. The main goal of the dataset *"is to facilitate research, analysis and smart explorative applications"* as told by Stefan Dietze[4].

The dataset is written in nTriples format. NTriples is a highly serialized version of the turtle RDF language[3], which is used in web semantics and structured ontologies. This dataset can be loaded into any database management system that supports RDF data. However the main stream graph databases do not support RDF format. Therefore we need to look into alternative formats. Cypher Text, a format used for querying Neo4j[1], an open-source graph database management system, is a very well known and documented format. Therefore the conversion of nTriples to Cypher is

the primary step in using Neo4j to load the LAK Dataset.

For converting from one format to another there are many design considerations that are needed to be taken note of. The most primitive design consideration is the choice of what scripting language is to be used. Such a choice affects the rest of the course of the project. A good choice would be a language that contains a very well defined string processing and manipulation capabilities. It must have the ability to match patterns based on given regular expressions (RegEx). This is because scripts used to convert to different formats require RegEx to match the input strings to a pre-defined alternative and then perform the action that is required for it. Our choice for the language was Python as it not only has very powerful tools for string processing operation but also packages for Neo4j, the backend database.

Once the database is up and running, the second challenge arises, which is how to query the database. As the Neo4j database accepts queries only in Cypher format we will have to write the queries in Cypher only.

With all the background work done, we now have to provide a user interface to the end user. This can be done either using a client program or through the default browser view. A client program is better suited as the work of the user will be greatly reduced. We can accept input in plain English format and then have another script file to convert the accepted data into Cypher query. Finally once the query is executed we provide a method to view only those attributes that user requires. The user must be able to also download a copy.

All the above things help provide better visualization of the LAK dataset which can be used by the users who want to extract data from the dataset.

## 2. EXTRACTION AND VISUALIZATION

This paper can be broadly split into two parts, firstly *Visualization* and secondly *Extraction of Data*. Visualization includes porting of the LAK-Dataset from the present format to another format which is Cypher text. Extraction involves having a visual front end module to query the database (so formed) and allow any user to have his requirements met.

### 2.1 Visualization of LAK-Dataset

The conversion of the dataset from ntriples involves a series of steps. We have to first *cleanse* the dataset, that is, remove the redundant triples. nTriples has inbuilt redundancy removal algorithms lacking in Cypher. The redundant triples can be conviniently deleted by adding the triple store to a set data-structure. Due to the nature of this data-structure the redundant triples are automatically removed and we get the *cleansed* dataset.

The second step is to segregate the triples to three different types. The triples are classified as nodes (those triples with `rdf#type` ontology), attributes (those triples with literal values as object) and relations (those triples with a URI as object). The segregation is done using a BASH script (as they are very efficient).

The conversion of a node with attributes can be achieved

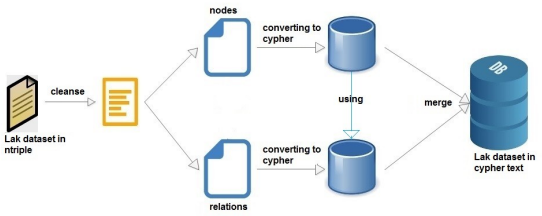


**Figure 1: Steps involved in the conversion of the LAK Dataset from nTriple format to Cypher text.**

with minimal overhead. This can be done using any scripting languages (We chose Python, due to its support for Regular Expressions). For e.g a triple of the form `<lak/person/abc> <22-rdf-syntax-ns#type> <foaf/0.1/Person>` can be written in Cypher text as `CREATE (a:Person)` and node attributes like `<lak/person/abc> <foaf/0.1/name> "abc"` can be added using `UPDATE` clause. However to maintain simplicity, we have added the attributes along with the `CREATE` statement (to reduce size we introduce a small overhead by using look-ahead). So the converted Cypher query can be written as `CREATE (a:Person{name = "abc"})`.

Finally the relations have to be converted to Cypher. This is the most challenging part as the input format is a structured dataset used for sematic web. The graph database of our choice is a flat graph database (so chosen to add to speed and reduce size). Thus the relationships cannot be directly modelled. The relationships can only be linked between two nodes and thus metadata links cannot be represented in Cypher (but it does not come in way as we are providing the end user, a method to extract information contained in the links and not about the links). To create relationships in Cypher text we will need the variables used to create the nodes. Therfore if one of the nodes is `CREATE (a:Person)` and the other is `CREATE (b:InProceedings)` and the relationship is `maker`, then we can create a relationship in Cypher text as `CREATE (a)-[:'maker']->(b)`, hence we have a relationship between two nodes in Cypher text. A small code is also required to convert unicode 32bit values (written with escape `\U` ) is to be converted to two surrogates, as Neo4j supports only unicode written in 16bit hex (written with escape `\u` ).

In this way, at first the nodes along with the attributes are converted to Cypher and then using the Cypher nodes, the relations are made. They both are then merged together to form the complete dataset in Cypher Text. The whole process can be summarised as shown in Figure 1.

### 2.2 Loading of Data

The converted LAK-Dataset in Cypher text can be loaded onto the Neo4j server either using the webclient or through any language API provided by Neo4j. However loading the dataset is not a straight forward technique as the variables need to be remembered (which is not possible using API as the statements are committed after every transction and variable names are lost). Thus we need to again seperate the dump into two that is nodes and relations. The nodes can be loaded directly as they do not need remembering variables

**Table 1: The final output if the user wants a list of all the labels and identifiers of authors who's name start's with "A."**

| Slno | identifier | label |
|---|---|---|
| 1 | person/a-a-von-davier | A. A. Von Davier |
| 2 | person/a-al-qaraghuli | A. Al-qaraghuli |
| 3 | person/a-altun | A. Altun |
| 4 | person/a-kharrufa | A. Kharrufa |
| 5 | person/a-muslim | A. Muslim |
| 6 | person/a-zapata-gonzalez | A. Zapata-gonzalez |

however the relations have to split apart.

Every relation statement in Cypher text is matched with the two variables it requires. The variables can be matched using `MATCH` clause. Thus to match variables, we introduce a seperate attribute to each of the nodes called as `identifier` which will contain the URI of the node. Therefore, for the relation `CREATE (a)-[:'maker']->(b)` we will first add two statements before the statement, which are `MATCH a WHERE a.identifier = "person/abc"` and `MATCH b WHERE b.identfier= "conference/abc/paper/xyz"`. These three statements are made into a single Cypher query and executed on the server. Thus by doing so, all the nodes and relations are loaded on to the database and it is now ready for the user input.

## 2.3 Extracting Data

Extraction of data is the process of presenting the data (in the database) to the user such that the data so presented is not only accurate to the user's needs but also easily readable. All the query statements, for data extraction, in Neo4j using Cypher have to use the `MATCH` clause. For example, if the user wants a list of all authors whoever has written any paper in the dataset. For this we can have a simple query, `MATCH a:Person RETURN a`. This will return the list in JsON. This format is very difficult to parse itself (and thus cannot be displayed as result). Thus on receiving the data from the server we convert it to CSV immediately and save it in a file called *export.csv*.The conversion is acheived by using the relevant library provided by Python . Using this library, we delete the un-necessary fields and convert only those fields required. The user then has to specify as to what all attributes is required (that is if only the `firstName` and the `lastName` of the authors is needed)[1] and the number of rows in the final output (or no upperbound). The user is also given an option to add conditons to the chosen attributes (that is the `firstName` value should be `"A. "` and so on). The final output is then displayed to the user as a table with the required values as shown in Table 1.

The user can now take the output as either a PDF document or a TXT file with the table[2]. As the program returns tabular data, it can be converted to any desirable form including SQL dump or LaTeXTable format.

**Table 2: The final output if the user wants a list of all the labels and identifiers of authors whohave written the paper "Edu-mining for Book Recommendation for Pupils"**

| Slno | identifier | label |
|---|---|---|
| 1 | person/junichi-kakegawa | Junichi Kakegawa |
| 2 | person/koji-suda | Koji Suda |
| 3 | person/keigo-takeda | Keigo Takeda |
| 4 | person/koichiro-morihiro | Koichiro Morihiro |
| 5 | person/ryo-nagata | Ryo Nagata |

In the same way the user can also ask for relationships. The user is provided with an option to return which ever node that is required. For example, the user may ask for a list of authors who have written a particular paper. The user then needs to enter the name of the paper . Then the user will get a table just like that with the nodes.

The complexity increases in the case that the user wants both the nodes as output. In this case the output will be a join of the two returned nodes [3].

## 2.4 User Interface

The user interface can either be a command line or a much more readable GUI. The GUI is done using Tkinter and CLI using BASH. The front end basically just accepts the inputs from the user and has a call back to the different programs at the backend. The GUI itself is called using a BASH script. The GUI, while taking inputs, does a few essential jobs. Firstly it will restrict any statements which may delete any of the data present in the database (to prevent Injection attacks). This is called input validation. Secondly it does another important job of forming the query. The Query is generated on the client end itself as it would not be efficient to send the details to a callback program which will genrate the query. The GUI also shows the user his required output in tabular format. Lastly it will direct the user to the location of the output files. (If gnome is present, the final output will pop-up). Whatever functionality is present in the GUI is also present in the CLI program. (Running the program `run` with a `-c` option launches the CLI tool, whereas doing that with a `-g` option launches the GUI).

A small interface can be seen in Figure 2 and 3.

# 3. CONCLUSIONS

Analytics is no more just an alternative for proof establishment, but has evolved over time to become a necessity in many fields including a complex process such as learning. Hence the research in the field of learning analytics has accelerated in the recent past due to the need for better results and overall improvement of the learning process. This field has not only gained much importance in the recent past but

[1]The number of attributes the user can choose is restricted to 4, due to the constraints in printing the table on a letter sized sheet.

[2]The intermediate CSV file created is not deleted as the user can also take a copy of the CSV, but this is highly not recommended as the CSV file is delimited using double quotes and the quote character is single quote. Hence it is illogical for the user to take the CSV file unless he will parse it using an automated CSV parser

[3]Again, due to the limit of the size of the paper, to fit in A4 sheet, the tables will be printed one after the other even though the User Interface shows them, side by side.

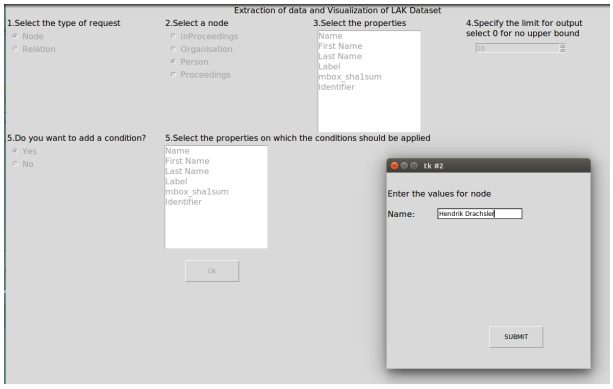


**Figure 2: The GUI window to select a node**

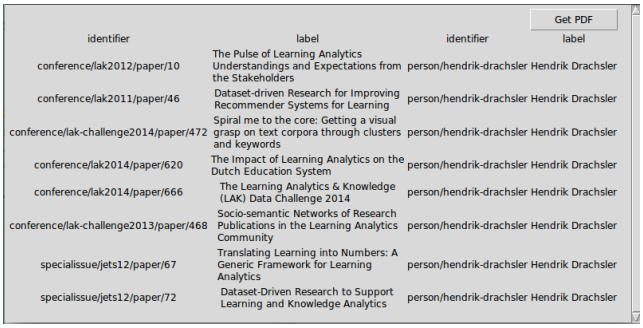


**Figure 3: Output window shown for the result of the list of papers written by Hendrik Drachsler**

it has also surpassed many other fields of analysis in terms of research publications.

For a better analysis and understanding of the process, an efficient way of collecting and organizing data had to be developed so that the research professional that will have to analyze the patterns to research in this field need not worry about the internals of the data storage and maintenance.

This work carried out caters to the need by creating a level of abstraction on the raw data that is LAK dataset. Hence the abstraction takes care of not dumping too much of details on the user of the dataset.

In addition to this, the graph database abstraction provided helps users to concentrate more on the analytical approach and not waste their time on the details of organization and maintenance of the dataset. The choice of the graph database contributes to easier maintenance of data and faster transaction of input and output. Neo4j assures the user that his data is not corrupted. Also there is an option for the user to archive the query which has been executed. Hence all care has been taken to make sure that the user is at most ease while he is researching in this field.

The choice of python as a scripting language for handling data conversions is justified by the ease of writing scripts and the large library it provides.

In conclusion, the title of the project *"A novel method for information extraction and visualization of the Learning Analytics and Knowledge dataset"* is well justified by the conversion of the dataset into a graph database to provide a better visualization and by providing a good means of data extraction by querying the so formed graph database.

# 4. MOTIVATION

This dataset can be used to create a graph database from the present dataset provided. The advantage of this is that better visualization techniques can be used (for example, the Neo4j web client can be used). As the dataset is now in Cypher text, further enhancements like adding extra links are now easier.

The main advantage of the visualization tools is that it reduces the amount of time required to extract viable information from the dataset. The user now need not spend any time on forming the query and then executing it. These tools give him an undue advantage with which he can not only visualize the data, but he can also take a copy of the results. The tabular format of the result has a unique advantage, it is very readable as compared to graphs. Also table formats are easier to understand when the results are printed out on a hard copy.

Thus this work carried out will be useful when there is a need to visualize the LAK Dataset and extract information in a very efficient way, with minimum time overhead.

# 5. ACKNOWLEDGMENTS

The authors would like to thank Prof D Jawahar, Prof Ajoy, Dr K N B Murthy and Dr K S Sridhar of PESIT, Bangalore for their support and encouragement. The authors are indebted to Ordell Ugo where this work was carried out.

We would also like to thank the Python software foundation and the Neo4j team for providing us with the required software. A special thanks to J Mark and M Needham of the Neo4j team for helping us in this work. We would like to thank A B Pillai for his code to convert a text file to PDF.

# 6. REFERENCES

[1] Neo4J Developers. Neo4j. *Graph NoSQL Database*, 2012.

[2] George Siemens and Phil Long. Penetrating the fog: Analytics in learning and education. *Educause Review*, 46(5), 2011.

[3] Tim Berners-Lee, David Beckett, Eric Prud hommeaux, and Gavin Carothers. Turtle-terse rdf triple language. *W3C team submission, W3C, Mar*, 20(11), 2009.

[4] Davide Taibi and Stefan Dietze. Fostering analytics on learning analytics research: the lak dataset. 2013.

# APPENDIX

# A. LINKS TO THE DATASET

## A.1 The Raw data dump

The complete data dump of the LAK Dataset in Cypher text can be found here.

## A.2 Visualization tools

The set of tools required to visualize the data-set is available here.