Deep Learning on Graph Database using Neo4j

Tushar Pahuja(15BCE1252)[1], Osho Agyeya(15BCE1326) [2], Hargur Bedi(15BCE1257)[3], Pushpit Bagga(15BCE1136)[4]

[1]*Vellore Institute of Technology, Chennai, India*
[2]*Vellore Institute of Technology, Chennai, India*
[3]*Vellore Institute of Technology, Chennai, India*
[4]*Vellore Institute of Technology, Chennai, India*

[1]*E-mail: tushar.pahuja2015@vit.ac.in*
[2]*E-mail: osho.agyeya2015@vit.ac.in*
[3]*E-mail: hargurpartap.singhbedi2015@vit.ac.in*
[4]*E-mail: pushpit.bagga2015@vit.ac.in*

Abstract

Neo4j is a graph database management system developed by Neo4j, Inc. Described by its developers as an ACID -compliant transactional database with native graph storage and processing, Neo4j is the most popular graph database according to DB-Engines ranking, and the 22nd most popular database overall. Keras is an open source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit or Theano. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. The project is aimed to connect a Neo4j graph database to Keras. The main objective of the project is to create a neural network for a review prediction task. The model is trained on the set of review scores given by the persons for products. The database is created in Neo4j and it is injected into Keras learning model to make predictions regarding reviews of products.

Literature Review

The following research papers and articles were analyzed and studied while doing the project.

1) ***Deep Learning in Neural Networks: An Overview*** by ***J¨ urgen Schmidhuber***
In recent years, deep artificial neural networks (including recurrent ones) have won numerous contests in pattern recognition and machine learning. This historical survey compactly summarizes relevant work, much of it from the previous millennium. Shallow and deep learners are distinguished by the depth of their credit assignment paths, which are chains of possibly learnable, causal links between actions and effects. I review deep supervised learning (also recapitulating the history of backpropagation), unsupervised learning, reinforcement learning & evolutionary computation, and indirect search for short programs encoding deep and large networks.

2) ***Literature review about Neo4j graph database as a feasible alternative for replacing RDBMS*** by ***Félix MelChor sANtos lóPez, eulogio guillerMo SanTos De LA Cruz***

This paper reviews the definition of Neo4j graph database, its architectural components and the mechanisms for manipulating the data. The first point gives an overview of Neo4j features, then the structure and how it records and retrieves data. The following section evaluates four research papers and the main findings, and each article shows a benchmarking between Neo4j and relational databases as MySQL and Postgres. After, the review criticizes how the previous experiments were carried out, focusing on the methodologies, the quality of the simulated data, the sample size and the IT infrastructure used. The literature review concludes with recommendations about how to achieve deeper and precise benchmarking of Neo4j with relational databases. It also highlights the importance of using real social web portals or information systems currently working in production environments. More research about the performance with the Delete and Update operations is required in order to claim that Neo4j is a realistic and strong candidate for replacing the relational databases that have been hoarding the market of enterprise applications.

3) ***Graph or Relational Databases: A Speed Comparison for Process Mining Algorithm*** by ***Jeevan Joishi, Ashish Sureka***
Process-Aware Information System (PAIS) are IT systems that manages, supports business processes and generate large event logs from execution of business processes. An event log is represented as a tuple of the form CaseID, Time Stamp, Activity and Actor. Process Mining is an emerging area of research that deals with the study and analysis of business processes based on event logs. Process Mining aims at analyzing event logs and discover business process models, enhance them or check for conformance with an a priori model. The large volume of event logs generated are stored in databases. Relational databases perform really well for certain class of applications. However, there are certain class of applications for which relational databases are not able to scale. A number of NoSQL databases have emerged to encounter the challenges of scalability. Discovering social network from event logs is one of the most challenging and important Process Mining tasks. Similar-Task and Sub-Contract algorithms are some of the most widely used Organizational Mining techniques. Our objective is to investigate which of the databases (Relational or Graph) perform better for Organizational Mining under Process Mining. An intersection of Process Mining and Graph Databases can be accomplished by modelling these Organizational Mining metrics with graph databases. We implement Similar-Task and Sub-Contract algorithms on relational and NoSQL (graph-oriented) databases using only query language constructs. We conduct empirical analysis on a large real-world data set to compare the performance of row-oriented database and NoSQL graph-oriented database. We benchmark performance factors like query execution time, CPU usage and disk/memory space usage for NoSQL graph-oriented database against row-oriented database.

4) ***A Review on Deep Learning*** by ***Yann LeCun, Yoshua Bengio & Geoffrey Hinton***
The research article mentions that Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional nets have brought about breakthroughs in processing images, video, speech and audio, whereas recurrent nets have shone light on sequential data such as text and speech.

5) ***A STUDY ON GRAPH STORAGE DATABASE OF NOSQL*** by ***Smita Agrawal and Atul Patel***
Big Data is used to store huge volume of both structured and unstructured data which is so large and is hard to process using current / traditional database tools and software technologies. The goal of Big Data Storage Management is to ensure a high level of data quality and availability for business intellect and big data analytics applications. Graph database which is not most popular NoSQL database compare to relational database yet but it is a most powerful NoSQL database which can handle large volume of data in very efficient way. It is very difficult to manage large volume of data using traditional technology. Data retrieval time may be more as per database size gets increase. As solution of that NoSQL databases are available. This paper describes what is big data storage management, dimensions of big data, types of data, what is structured and unstructured data, what is NoSQL database, types of NoSQL database, basic structure of graph database, advantages, disadvantages and application area and comparison of various graph database.

Proposed Work

The main aim of the project is to study the relationship between the people and the reviews of products written by them. The main challenge of the project is to predict **what review score a person will give to a product**.
In order to study the relationship, a simple product review graph would be generated. The graph would have the following nodes with properties:

- **Person:** with a style_preference vector of width 6 that one-hot encodes with product style they like.
- **Product:** with a style vector of width 6 that one-hot encodes which style that product is.
- **Review:** between **person** and **product**, which a score floating point number.

How review scored are calculated?

Review score are calculated as the dot product between a person's style_preference and a product's style.

For example, a person with style_preference [0,1,0,0,0,0] reviews product Nintendo Switch with style [0,1,0,0,0,0], and her review score is 1.0. When she reviews a PS4, with style [1,0,0,0,0,0], her review score is 0.0.

System Architecture

A Deep Learning Neural Network model is used for this project. The model has two dense layers, of width 6, with tanh activation layers. These give it room to separately combine the corresponding elements from the style preference of the persons and style of the movie vectors. Then an output layer of width one is applied, also with tanh activation. The network at this point has generated a single value, the prediction for the review score. With our model built, we compile it to use the popular Adam optimizer and mean squared error as the loss function.

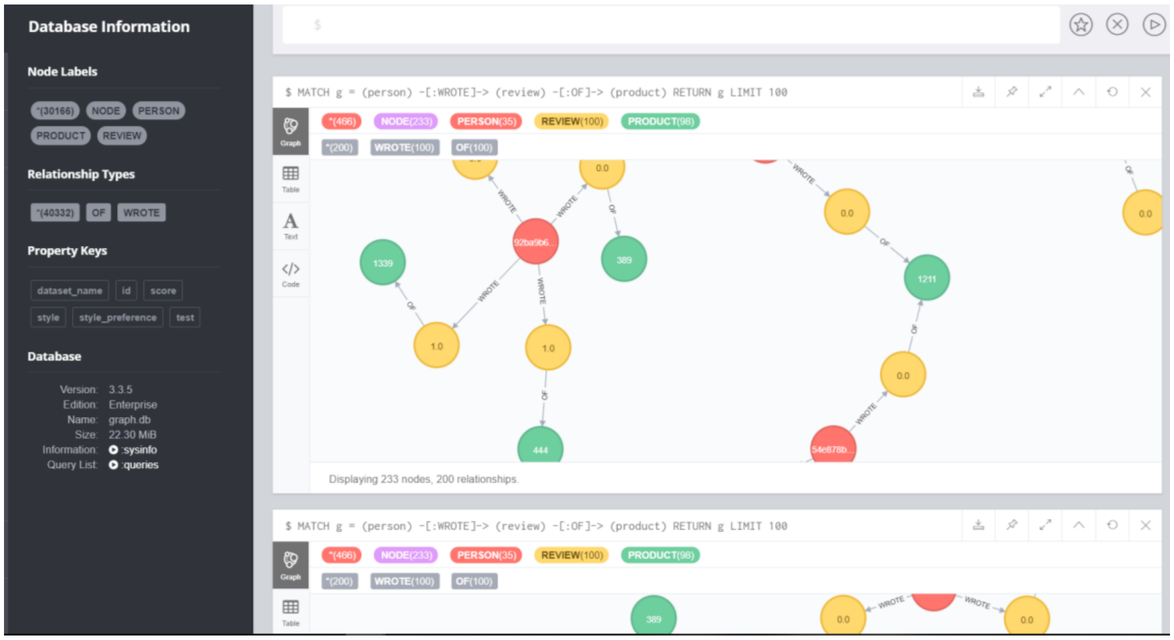
Software Used:

Neo4j, Python, Cypher, Keras

Hardware Used:

ASUS GL552VW ROG, NVIDIA GeForce GTX 960M with Compute capability 6 running Windows Operating System

System Diagram



```
$ MATCH g = (person) -[:WROTE]-> (review) -[:OF]-> (product) RETURN g LIMIT 100
```

Displaying 233 nodes, 200 relationships.

```
$ MATCH g = (person) -[:WROTE]-> (review) -[:OF]-> (product) RETURN g LIMIT 100
```

Version: 3.3.5
Edition: Enterprise
Name: graph.db
Size: 22.30 MiB
Information: :sysinfo
Query List: :queries

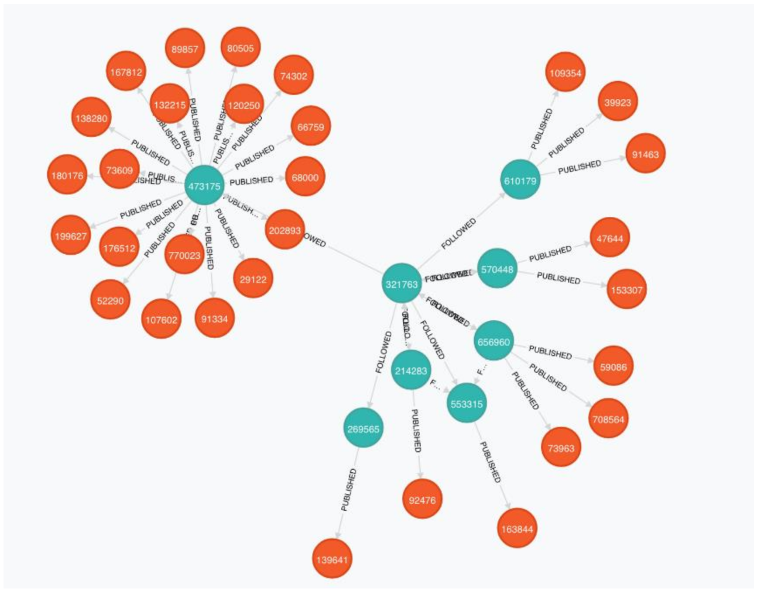
Modules

1. Creation of graph database manually in Neo4j

This involves converting structured database into graph database. When deriving a graph model from a relational model, we should keep the following guidelines in mind:

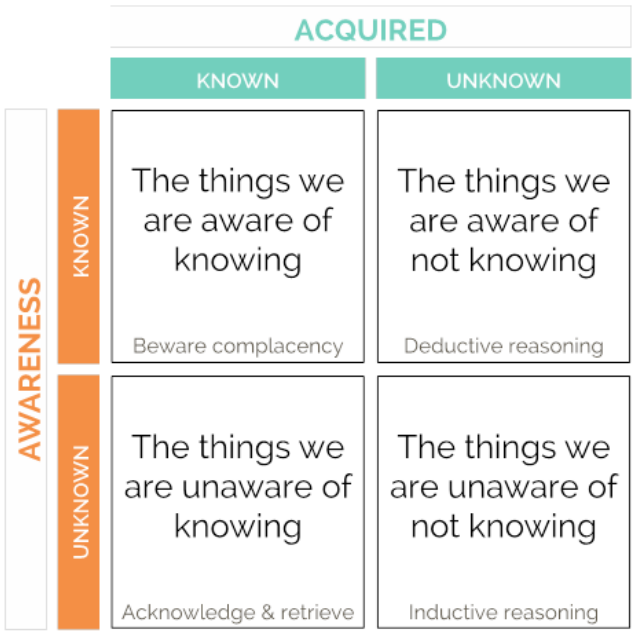
A row is a node

A table name is a label name



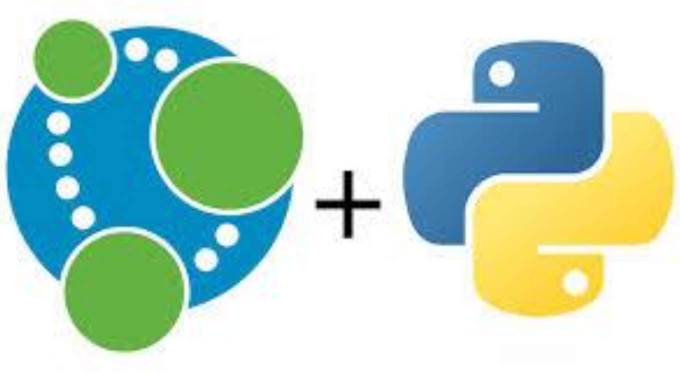
2. Exploratory data analysis

In statistics, exploratory data analysis (EDA) is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. This involves studying the data to find patterns in it in order to find clusters and related data.


ACQUIRED
KNOWN
UNKNOWN
AWARENESS
KNOWN
UNKNOWN
The things we are aware of knowing
Beware complacency
The things we are aware of not knowing
Deductive reasoning
The things we are unaware of knowing
Acknowledge & retrieve
The things we are unaware of not knowing
Inductive reasoning

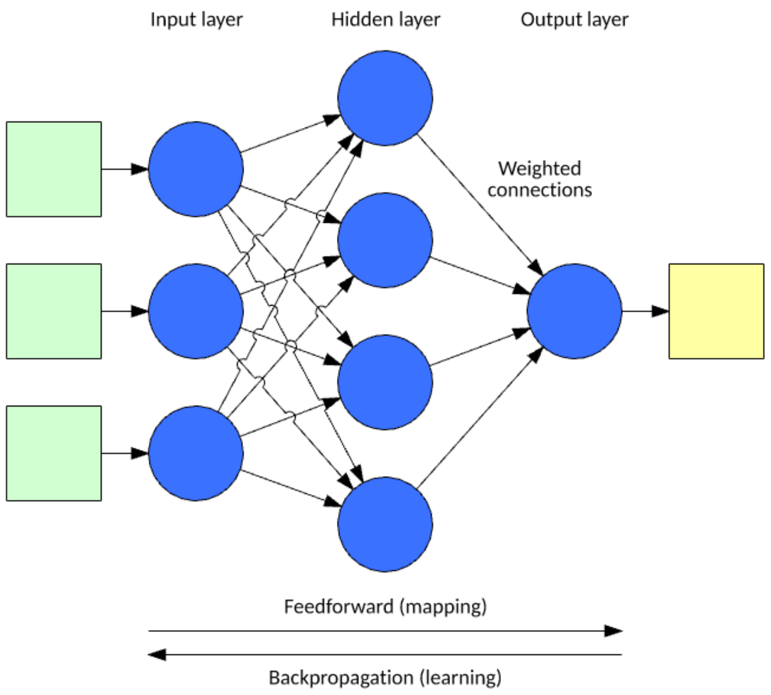
3. Establishing connection between Neo4j and Keras

After having studied the data, we are supposed to connect the graph database to the Keras code which we want to execute by the usage of Neo4j APIs.



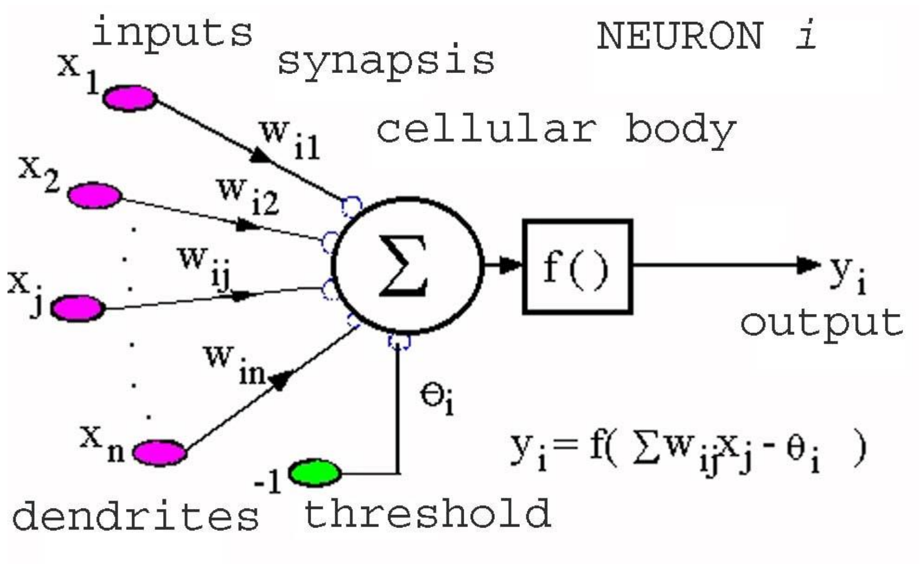
4. Create a machine learning model

This involves defining the layers of the neural network along with the loss function and optimizer. The no. of layers in the output layer varies according to the input statement.

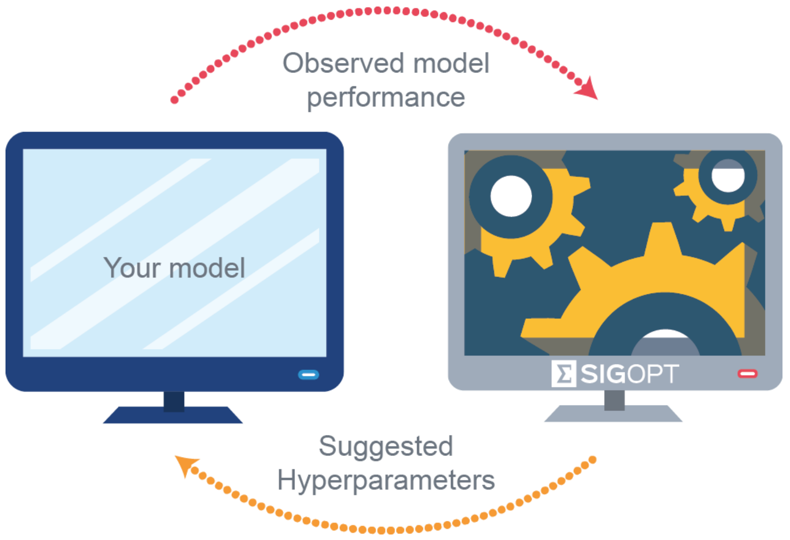


5. Training session of the model

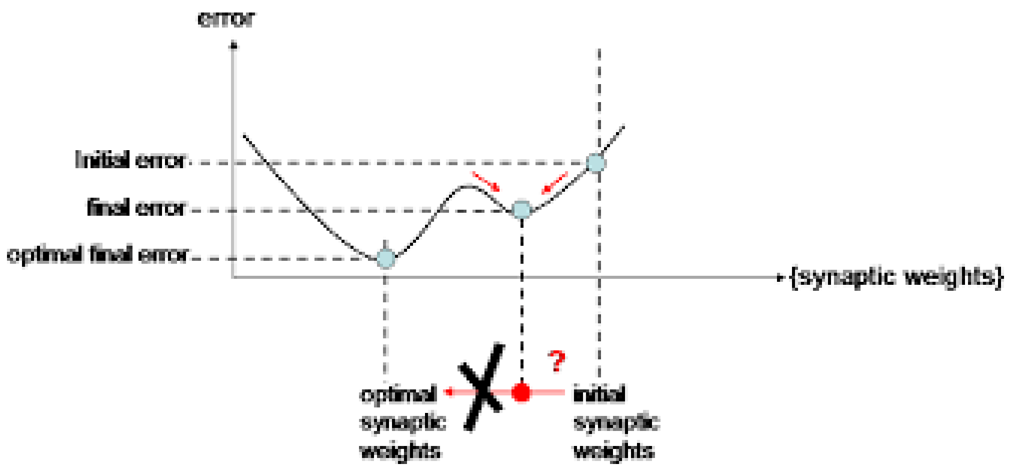
This involves changes in the initial values of the network according to the prediction and true value. The neurons try to adjust themselves according to the data that it received and predicted output.



6. Hyperparameter tuning after observing results of training session
The way in which the neural network trains itself goes on to decide whether the value of the learning rate, no of layers, no. of neurons in each layer should be changed or not to cause faster converging.

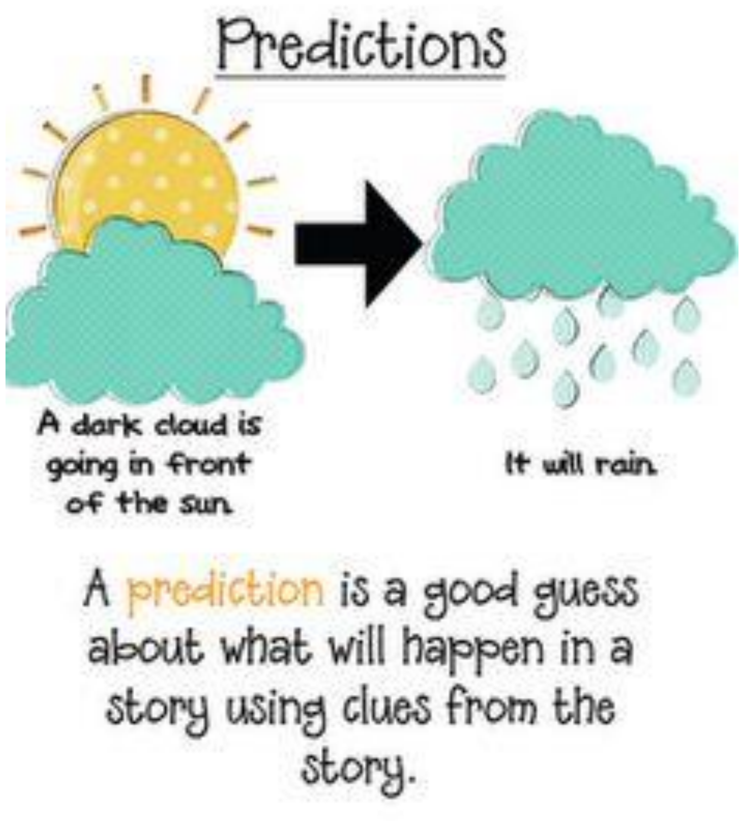


7. Finalizing the model
After finalizing the values of the hyperparameters, the neural network is finalized and retrained from scratch.

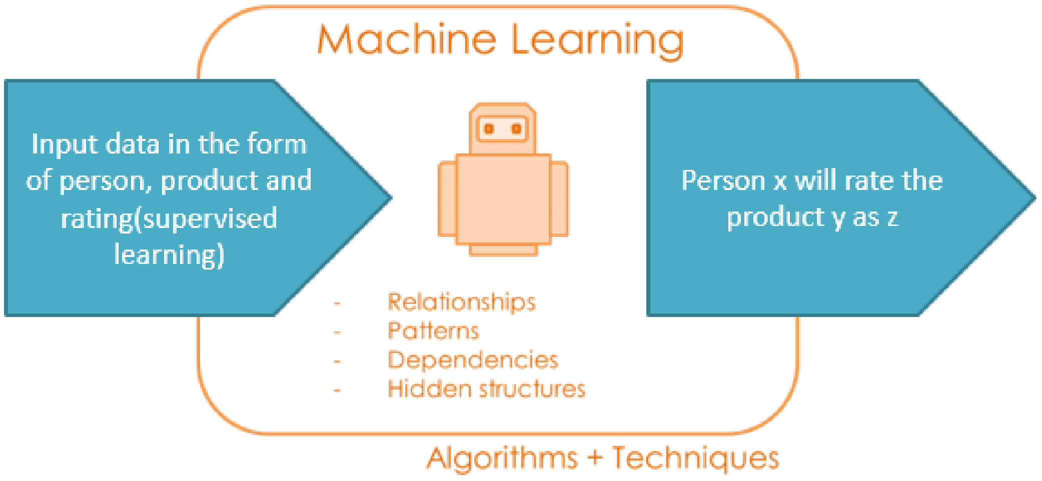


8. Making final predictions

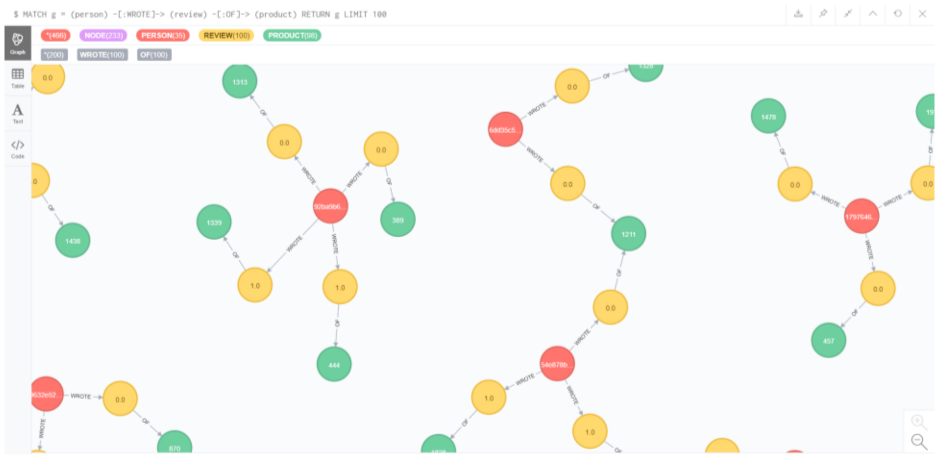
Final predictions are made for unseen data and these predictions are stored for later use.


Predictions
A dark cloud is going in front of the sun
It will rain.
A prediction is a good guess about what will happen in a story using clues from the story.

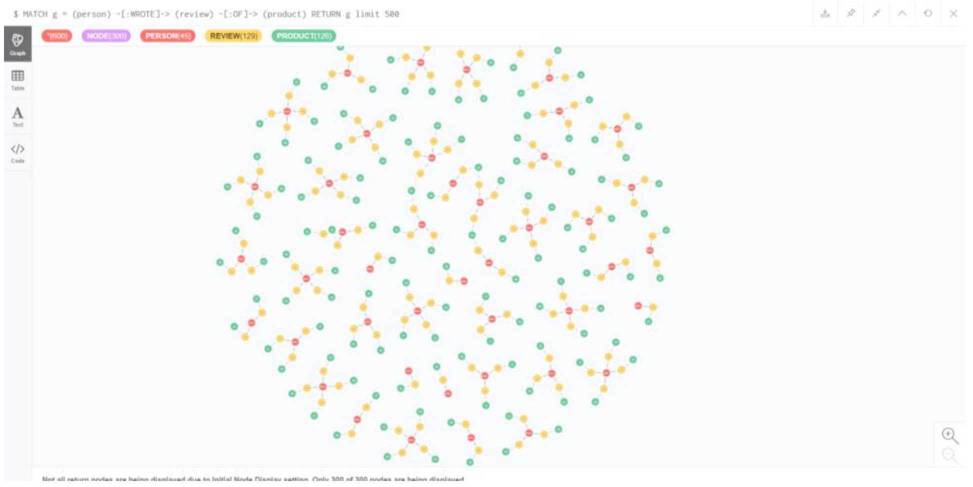
The model is supposed to predict the **review score a person will give to a product** using the knowledge learnt from the dataset.


Machine Learning
Input data in the form of person, product and rating(supervised learning)
Person x will rate the product y as z
- Relationships
- Patterns
- Dependencies
- Hidden structures
Algorithms + Techniques

Visual representation of Database



```
$ MATCH g = (person) -[:WROTE]-> (review) -[:OF]-> (product) RETURN g LIMIT 100
```



```
$ MATCH g = (person) -[:WROTE]-> (review) -[:OF]-> (product) RETURN g limit 500
```

Not all return nodes are being displayed due to Initial Node Display setting. Only 300 of 300 nodes are being displayed

Code

train.py

```
#!/usr/bin/env python3

from keras.models import Sequential
from keras.layers import Dense

from graph_sequence import *

def generate_model():
	model = Sequential([
		Dense(6,
			input_shape=(12,),
			activation='tanh'),
		Dense(6, activation='tanh'),
		Dense(1, activation='tanh'),
	])

	return model

def train(args):

	model = generate_model()

	model.compile(loss='mean_squared_error',
	              optimizer='adam',
	              metrics=['accuracy'])

	seq_train = GraphSequence(args)
	model.fit_generator(seq_train, epochs=10)

	seq_test = GraphSequence(args, test=True)
	result = model.evaluate_generator(seq_test)

	final_res=round(result[1]*100)

	print("Accuracy=",final_res)

if __name__ == '__main__':
     import argparse
     parser = argparse.ArgumentParser()
     parser.add_argument('--database', type=str, default="hosted")
     args = parser.parse_args()

     train(args)
```

graph_sequence.py

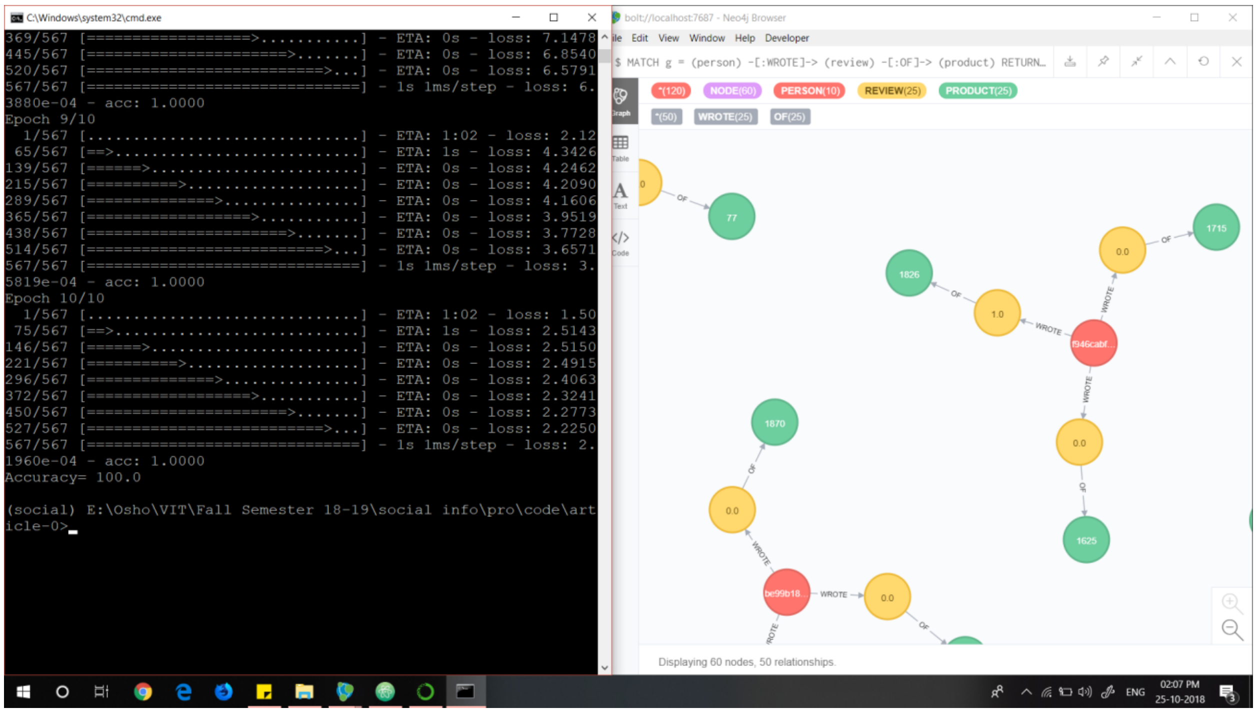
```
import keras
import numpy as np
import more_itertools
import json
from neo4j.v1 import GraphDatabase, Driver

class GraphSequence(keras.utils.Sequence):

     def __init__(self, args, batch_size=32, test=False):
          self.batch_size = batch_size

          self.query = """
               MATCH p=
                    (person:PERSON)
                         -[:WROTE]->
                    (review:REVIEW      {dataset_name:{dataset_name},
test:{test}})
                         -[:OF]->
                    (product:PRODUCT)
               RETURN  person.style_preference  +  product.style  as  x,
review.score as y
          """

          self.query_params = {
               "dataset_name": "article_0",
               "test": test
          }

          with open('./settings.json') as f:
               self.settings = json.load(f)[args.database]

          driver = GraphDatabase.driver(
               self.settings["neo4j_url"],
               auth=(self.settings["neo4j_user"],
self.settings["neo4j_password"]))

          with driver.session() as session:
               data              =              session.run(self.query,
**self.query_params).data()
               data = [ (np.array(i["x"]), i["y"]) for i in data]

               # Split the data up into "batches"
               data = more_itertools.chunked(data, self.batch_size)

               # Format our batches in the way Keras expects them:
               # An array of tuples (x_batch, y_batch)

               # An x_batch is a numpy array of shape (batch_size, 12),
               # containing the concatenated style and style_preference
vectors.

               # A y_batch is a numpy array of shape (batch_size,1)
containing the review scores.

               self.data  =  [  (np.array([j[0]  for  j  in  i]),
np.array([j[1] for j in i])) for i in data]


     def __len__(self):
          return len(self.data)

     def __getitem__(self, idx):
          return self.data[idx]
```

settings.json

```
{
      "hosted": {
            "neo4j_url": "bolt://b2355e7e.databases.neo4j.io",
            "neo4j_user": "readonly",
            "neo4j_password": "OS5jLkVsOUZCVTdQOU5PazBo"
      },
      "local": {
            "neo4j_url": "bolt://localhost",
          "neo4j_user": "neo4j",
          "neo4j_password": "neo4j"
      }
}
```

Results and Discussion

The given neural network was trained on the data defined. The nodes have been tagged as test=" false" to represent those nodes which are supposed to be used for training and test=" true" to represent nodes that are supposed to be used for testing. The test dataset is an unseen dataset that is used to calculate the final value of accuracy.



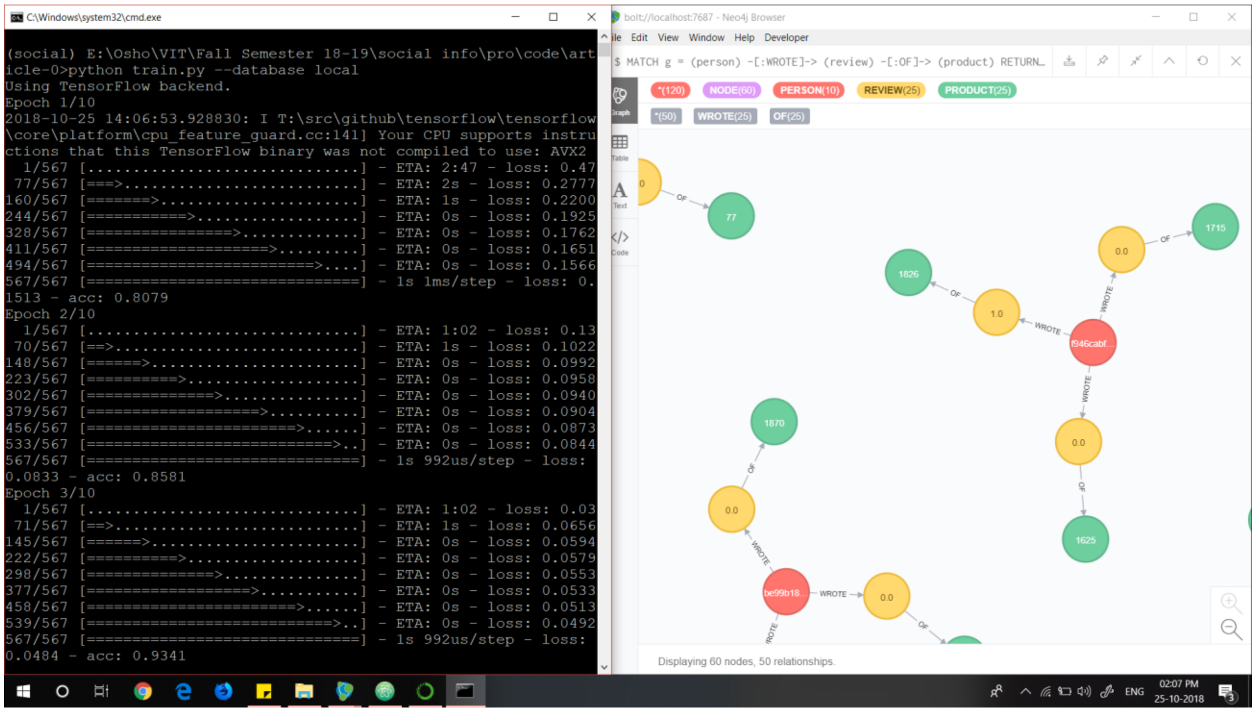
```
369/567 [==================>...........] - ETA: 0s - loss: 7.1478
445/567 [======================>.......] - ETA: 0s - loss: 6.8540
520/567 [==========================>...] - ETA: 0s - loss: 6.5791
567/567 [==============================] - 1s 1ms/step - loss: 6.
3880e-04 - acc: 1.0000
Epoch 9/10
  1/567 [..............................] - ETA: 1:02 - loss: 2.12
 65/567 [==>...........................] - ETA: 1s - loss: 4.3426
139/567 [======>.......................] - ETA: 0s - loss: 4.2462
215/567 [==========>...................] - ETA: 0s - loss: 4.2090
289/567 [==============>...............] - ETA: 0s - loss: 4.1606
365/567 [==================>...........] - ETA: 0s - loss: 3.9519
438/567 [======================>.......] - ETA: 0s - loss: 3.7728
514/567 [==========================>...] - ETA: 0s - loss: 3.6571
567/567 [==============================] - 1s 1ms/step - loss: 3.
5819e-04 - acc: 1.0000
Epoch 10/10
  1/567 [..............................] - ETA: 1:02 - loss: 1.50
 75/567 [==>...........................] - ETA: 1s - loss: 2.5143
146/567 [======>.......................] - ETA: 0s - loss: 2.5150
221/567 [==========>...................] - ETA: 0s - loss: 2.4915
296/567 [==============>...............] - ETA: 0s - loss: 2.4063
372/567 [==================>...........] - ETA: 0s - loss: 2.3241
450/567 [======================>.......] - ETA: 0s - loss: 2.2773
527/567 [==========================>...] - ETA: 0s - loss: 2.2250
567/567 [==============================] - 1s 1ms/step - loss: 2.
1960e-04 - acc: 1.0000
Accuracy= 100.0

(social) E:\Osho\VIT\Fall Semester 18-19\social info\pro\code\art
icle-0>_
```

```
$ MATCH g = (person) -[:WROTE]-> (review) -[:OF]-> (product) RETURN…
```

C:\Windows\system32\cmd.exe
bolt://localhost:7687 - Neo4j Browser
ile Edit View Window Help Developer
*(120) NODE(60) PERSON(10) REVIEW(25) PRODUCT(25)
*(50) WROTE(25) OF(25)
Graph
Table
Text
Code
77
1715
1826
1.0
0.0
f946cabf...
1870
1625
be99b18
WROTE
OF
Displaying 60 nodes, 50 relationships.
ENG
02:07 PM
25-10-2018

The screenshot below shows the result of training and testing the neural network on the final value of hyperparameters.

The training is done for 10 epochs. Each epoch takes nearly a minute. The training accuracy increases gradually from 0.8079 to 1.0 after 5 epochs and the final testing accuracy is 1.0 on the testing set. The neural network successfully finds out the mathematical relationship of dot product of person_style and product_style being equal to final review score.



```
(social) E:\Osho\VIT\Fall Semester 18-19\social info\pro\code\art
icle-0>python train.py --database local
Using TensorFlow backend.
Epoch 1/10
2018-10-25 14:06:53.928830: I T:\src\github\tensorflow\tensorflow
\core\platform\cpu_feature_guard.cc:141] Your CPU supports instru
ctions that this TensorFlow binary was not compiled to use: AVX2
  1/567 [..............................] - ETA: 2:47 - loss: 0.47
 77/567 [===>..........................] - ETA: 2s - loss: 0.2777
160/567 [=======>......................] - ETA: 1s - loss: 0.2200
244/567 [===========>..................] - ETA: 0s - loss: 0.1925
328/567 [================>.............] - ETA: 0s - loss: 0.1762
411/567 [====================>.........] - ETA: 0s - loss: 0.1651
494/567 [=========================>....] - ETA: 0s - loss: 0.1566
567/567 [==============================] - 1s 1ms/step - loss: 0.
1513 - acc: 0.8079
Epoch 2/10
  1/567 [..............................] - ETA: 1:02 - loss: 0.13
 70/567 [==>...........................] - ETA: 1s - loss: 0.1022
148/567 [======>.......................] - ETA: 0s - loss: 0.0992
223/567 [==========>...................] - ETA: 0s - loss: 0.0958
302/567 [==============>...............] - ETA: 0s - loss: 0.0940
379/567 [===================>..........] - ETA: 0s - loss: 0.0904
456/567 [=======================>......] - ETA: 0s - loss: 0.0873
533/567 [===========================>..] - ETA: 0s - loss: 0.0844
567/567 [==============================] - 1s 992us/step - loss:
0.0833 - acc: 0.8581
Epoch 3/10
  1/567 [..............................] - ETA: 1:02 - loss: 0.03
 71/567 [==>...........................] - ETA: 1s - loss: 0.0656
145/567 [======>.......................] - ETA: 0s - loss: 0.0594
222/567 [==========>...................] - ETA: 0s - loss: 0.0579
298/567 [==============>...............] - ETA: 0s - loss: 0.0553
377/567 [==================>...........] - ETA: 0s - loss: 0.0533
458/567 [=======================>......] - ETA: 0s - loss: 0.0513
539/567 [===========================>..] - ETA: 0s - loss: 0.0492
567/567 [==============================] - 1s 992us/step - loss:
0.0484 - acc: 0.9341
```

```
$ MATCH g = (person) -[:WROTE]-> (review) -[:OF]-> (product) RETURN…
```

C:\Windows\system32\cmd.exe
bolt://localhost:7687 - Neo4j Browser
File Edit View Window Help Developer
*(120) NODE(60) PERSON(10) REVIEW(25) PRODUCT(25)
*(50) WROTE(25) OF(25)
Graph
Table
Text
Code
Displaying 60 nodes, 50 relationships.
02:07 PM
25-10-2018
ENG

References

[1] Graph or Relational Databases: A Speed Comparison for Process Mining Algorithm
Jeevan Joishi Ashish Sureka Indraprastha Institute of Information Technology, Delhi (IIITD)
New Delhi, India ABB Corporate Research Bangalore, India

[2] International Journal on Soft Computing, Artificial Intelligence and Applications (IJSCAI), Vol.5, No.1, February 2016 A STUDY ON GRAPH STORAGE DATABASE OF NOSQL
Smita Agrawal and Atul Patel
CSE Department, Institute of Technology, Nirma University, Ahmedabad
Dean of CMPICA, CHARUSAT University, Changa

[3] Deep Learning in Neural Networks: An Overview
Technical Report IDSIA-03-14 / arXiv:1404.7828 v4 [cs.NE] (88 pages, 888 references)
J¨urgen Schmidhuber The Swiss AI Lab IDSIA Istituto Dalle Molle di Studi sull'Intelligenza Artificiale University of Lugano & SUPSI Galleria 2, 6928 Manno-Lugano
Switzerland

8 October 2014

[4] Deep learning Yann LeCun, Yoshua Bengio & Geoffrey Hinton, doi:10.1038/nature14539

[5] https://neo4j.com/docs/developer-manual/3.4/

[6] https://keras.io/

[7]Python Machine Learning by Sebastian Raschka

[8]Neural Networks and Deep Learning by Coursera.com