Open Data Project

Jorge Vieira, Henry Qiu, Goktug Cengiz

June 2019

Task 1. Business idea and identification of two data sources

- Purpose statement

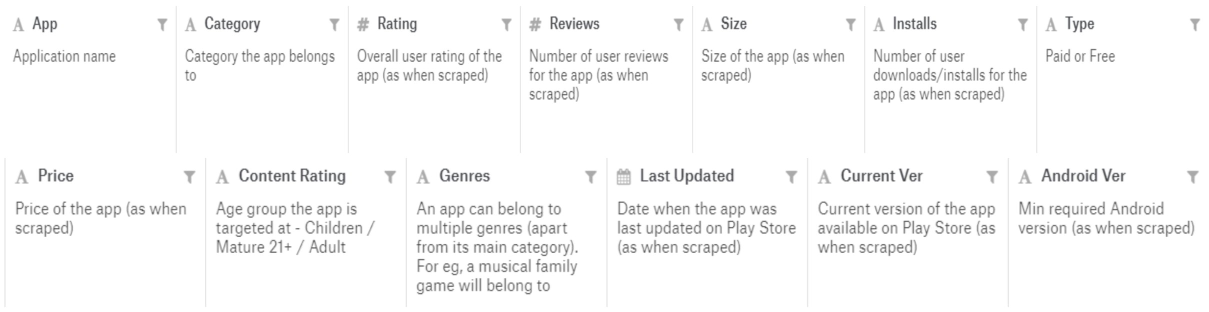
The aim of our project is to build an application recommendation system. It is very usual that software stores such as *google play, app store*, and so on, recommend applications to users based on the interests of the users. Nowadays recommendation systems appear in everywhere, and thanks to these systems the probability of the users consuming more products offered has been hugely increased. Applied to software applications, a good recommender can increase the number of downloads of the applications, this will help user to discover new needs and good applications that they unknew on one hand, and also a chance for the applications to gain users.

- Data sources

The data sources come from a *Kaggle* repository (link: https://www.kaggle.com/lava18/google-play-store-apps/downloads/google-play-store-apps.zip/6). It consists in a google play store applications repository, it contains nearly 10 kb of data collected for the purpose of analysing the android market.

Our data sources consists in two datasets: googleplaystore.csv, that contains the features and information of the applications, and googleplaystore_user_reviews.csv, that represents the reviews of the users after using some applications. Thanks to this information, on one hand, we will be able to do recommendations with the given interests of the users, analysing the similarity of his interests with the features of the applications, and on the other hand we will be able to do recommendations based on the opinions of the users in general to evaluate how good are the applications.

The format of the source files are csv, this means data are represented in tables. In the first dataset we can identify 13 columns:

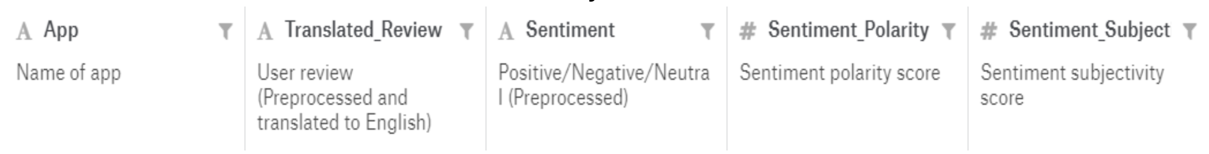


App	Category	Rating	Reviews	Size	Installs	Type
Application name	Category the app belongs to	Overall user rating of the app (as when scraped)	Number of user reviews for the app (as when scraped)	Size of the app (as when scraped)	Number of user downloads/installs for the app (as when scraped)	Paid or Free

Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
Price of the app (as when scraped)	Age group the app is targeted at - Children / Mature 21+ / Adult	An app can belong to multiple genres (apart from its main category). For eg, a musical family game will belong to	Date when the app was last updated on Play Store (as when scraped)	Current version of the app available on Play Store (as when scraped)	Min required Android version (as when scraped)

Among these attributes, we are interested in the app name, the category the app belongs to, the rating that users give, the number of downloads/installs, whether the app is paid or free and the genres or category.

For the second dataset we can identify 5 columns:



App	Translated_Review	Sentiment	Sentiment_Polarity	Sentiment_Subject
Name of app	User review (Preprocessed and translated to English)	Positive/Negative/Neutral (Preprocessed)	Sentiment polarity score	Sentiment subjectivity score

Among these attributes, we are interested in the app name, the sentiment (whether is positive, negative or neutral), and the sentiment subjectivity score.

The attributes we have removed are mainly because we think some of them contain redundant information (such as the genres), and some of them are not useful for the recommendation process (such as the size of the application, version, and so on).

- Graph family for the problem

The family of graph we are going to work with is property graph. We have seen that our model is very simple, so there is no need to build a semantic system. Taking into account that the model is simple, using a property graph makes much easier the creation of relations between the concepts. What is more, we are dealing frequently with attributes, property graph gives us much more simplicity when we need to define attributes of a class.

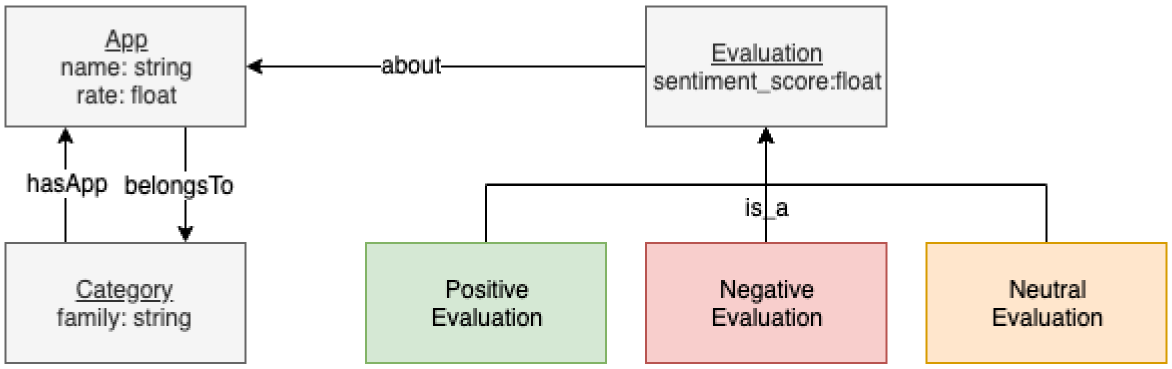
This means that the interface we are going to use to build our recommender system will neo4j, which means that the language we are going to use will be cipher. Thanks to the knowledge acquired in the laboratory sessions we can now apply the concepts to the practice.

Task 2: Creation of the integrated repository and definition of the integration / global schema

- Repository

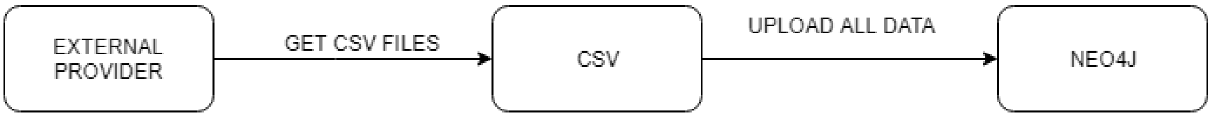
Regarding to the repository to where we are going to build the system, we will use Neo4J. Neo4J is a platform that allows us to deal with property graphs, it provides us methods to load the data into the repository, then it builds property graphs with the information provided, what is more, it works with cypher, that provide us to do queries on the graph using pattern matching.

- Integration / Global schema



We can see that in the schema of our project there are three main classes: App that represents the application, containing as attribute the name of the application (that is a string and has role of identifier of the application) and the rate of that application (that will has numerical value), Category that represents the kind or family that an application belongs to that only contains a string attribute that identifies a category, and an abstract class Evaluation that represents how the application is evaluated, we can see there are three subclasses: Positive Evaluation, Negative Evaluation and Neutral Evaluation, that identifies all the positive, negative and neutral evaluations respectively.

- Design of data flows



The sources that we found give us the data in the format .csv, therefore we have to load them with the function 'LOAD CSV' from neo4j.
The CSV information is distributed in 2 files, thus we have to merge both in the same graph. We use the application name to connect between both files.
As for the commands we used to load these files are:

- Metadata to automate exploitation process

Regarding to the use of metadata, we can add a new variable called Success as property of the class App, this will be a binary attribute that indicates whether the recommendation have been successful or not, that means the user has downloaded the application recommended or not.

This attribute will be used to future evaluations, especially data mining and machine learning algorithms. After someone uses an application, they can give a rate and satisfaction, as well as the genre. Those properties might help help recommending a new application similar to this one. The precision of the recommended application will increase with the number of “reviews” that the user provides after using that application.

Task 3: Exploitation idea. Goals and algorithms

- Specific algorithms and justification of graph

The main advantage of graphs is that over the graph we can perform multiple queries at different levels, in our case for example having categories as a class, if we want to do queries or analysis of the applications by categories we will perform it much better than a relational database: if we want all the applications of category “Arts”, we will have pointers to the category Arts and then all the nodes of applications pointed by that category, however in a relational database we would need to perform a scan of the hole table to identify all the applications of that category.

- Steps

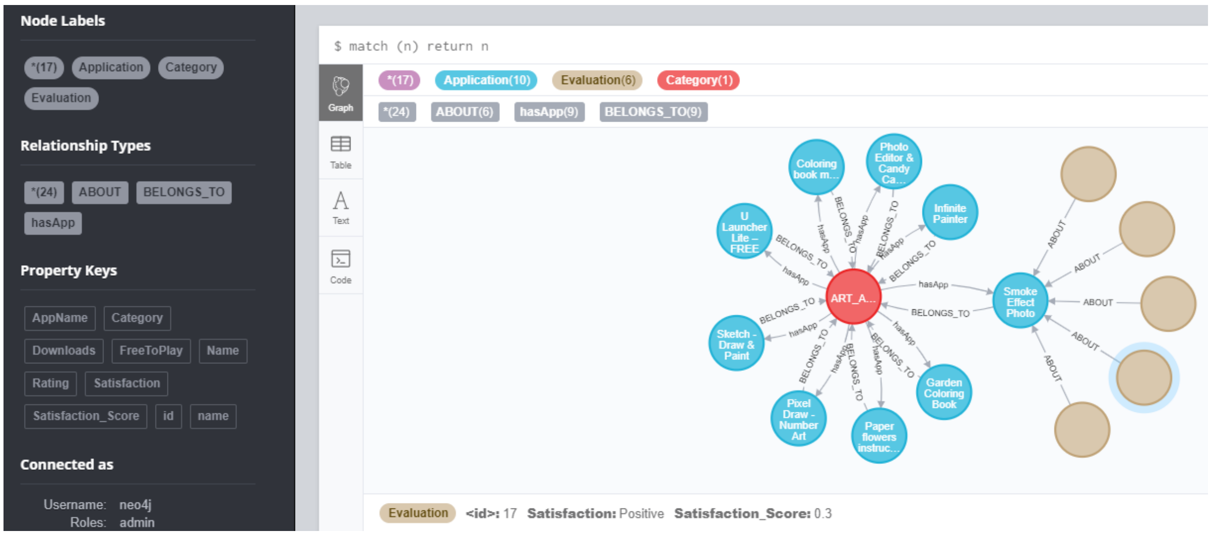
```
LOAD CSV WITH HEADERS FROM "file:///googleplaystorenew.csv" AS line
MERGE (category:Category {Category: line.Category})
CREATE (app:Application {AppName: line.App, Rating: line.Rating, Downloads:
line.Installs , FreeToPlay: line.Type, Category: line.Genres})
CREATE (app)-[:BELONGS_TO]->(category)
CREATE (category)-[:hasApp]->(app)

LOAD CSV WITH HEADERS FROM "file:///googleplaystore2new.csv" AS line
MERGE (app:Application {AppName: line.App})
CREATE (eva:Evaluation{Satisfaction:line.Sentiment,
Satisfaction_Score:line.Sentiment_Subjectivity})
CREATE (eva)-[:ABOUT]->(app)
```

We first load the data from the data source, while we load it we will be integrating the data into the system, because we will be choosing only the attributes we want and we will be passing from tables structure into graph structure.

Techniques such as component analysis can be used to get the correlation of the other variables and the target that will be in this case the variable Success, in this way we will be able to see which factors can determine whether the user will download the recommended app.

Other techniques that can be implemented are association rules, where we can know when users download the recommended applications which other applications they download.

Prove of Concepts


Node Labels

*(17) Application Category Evaluation

Relationship Types

*(24) ABOUT BELONGS_TO hasApp

Property Keys

AppName Category Downloads FreeToPlay Name Rating Satisfaction Satisfaction_Score id name

Connected as

Username: neo4j
Roles: admin

```
$ match (n) return n
```

*(17) Application(10) Evaluation(6) Category(1)

*(24) ABOUT(6) hasApp(9) BELONGS_TO(9)

Evaluation <id>: 17 Satisfaction: Positive Satisfaction_Score: 0.3

We have loaded our data into Neo4J, as the data was too bing, for better showing purposes, we have select very few amount of data to show the graph created. Here we can see the instances of apps represented by blue nodes, categories represented by red nodes, and evaluations represented by the brown ones. We can see also the relations between them. This is a prove that we know how to introduce the data into the system correctly choosing only the needed attributes.