University Of Crete
Department of Computer Science

Feature Extraction for Multi-layer Graph Link Prediction

Dimitrios Andreas Vogiatzidakis

Supervisor: Polyvios Pratikakis

A thesis submitted in partial fulfilment of the University's requirements for the degree of Bachelor of Science

June 28, 2020

Abstract

Data sets can naturally be represented as graph where nodes represent instances and links represent relationships between those instances.A fundamental issue with these types of data is that links may incorrectly exist between unrelated nodes or that links may be missing between two related nodes. The goal of link prediction is to predict those irregularities between the nodes of the graph and also possible future links.
This thesis tries to reproduce the feature extraction algorithm of TwitterMancer[1] using the Apache Flink framework. Apache Flink is a framework and distributed processing engine for stateful computations over unbounded and bounded data streams, designed to run in all common cluster environments and perform computations at in-memory speed and at any scale. We will use this framework to speedup the process of feature extraction and as a result, analyze larger scale graphs.
Using a data set of twitter interactions (follow, retweet, reply, quote) we find that we can achieve great speedup as we increase the number of cores of our cluster. That means the project can be scaled by simply adding more machines to the cluster. Substantially decreasing the time required to extract features from a data set , will result to faster and better results on link prediction.

Contents

Chapter 1

Introduction

Historically parallel computing was used for the simulation of scientific problems, particularly in the natural and engineering sciences, such as meteorology. This led to the design of parallel hardware and software, as well as high performance computing. This evolution allowed scientists to use larger data sets that previously couldn't.

1.1 Aims and Objectives

We are using Apache-Flink framework for parallel computation in order to extract features using larger Datasets and maybe improving the accuracy of the results of TwitterMancer.

1.2 Challenges

One of our main concerns was regarding the DRAM as our algorithm will produce a cross-product of N nodes, that we will later use to extract the features. Luckily, Flink is implemenented in a way to make full use of the available memory without ever exceeding it and causing errors.
Another issue was with our filesystem memory as the output of our algorithm is a .csv document containing all features for each pair of nodes(edge). Thus we have a file containting "N*N*Number of Features" records taking a lot of space on our hard drives. We partially solved this for testing purposes using datasets that will output the maximum available size and not more.

Chapter 2

Background

2.1 Theory Overview

In this thesis we are trying to extract features from a given dataset in order to perform a Logistic Regression and predict missing or future links.

2.1.1 Feature Extraction

In machine learning, pattern recognition and in image processing, feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations. Feature extraction is related to dimensionality reduction.

When the input data to an algorithm is too large to be processed and it is suspected to be redundant (e.g. the same measurement in both feet and meters, or the repetitiveness of images presented as pixels), then it can be transformed into a reduced set of features (also named a feature vector). Determining a subset of the initial features is called feature selection.[1] The selected features are expected to contain the relevant information from the input data, so that the desired task can be performed by using this reduced representation instead of the complete initial data. (source: wiki)

2.1.2 Logistic Regression

Logistic Regression, also known as Logit Regression or Logit Model, is a mathematical model used in statistics to estimate (guess) the probability of an event occurring having been given some previous data. Logistic Regression works with binary data, where either the event happens (1) or the event does not happen (0). So given some feature x it tries to find out whether some event y happens or not. So y can either be 0 or 1. In the case where the event happens, y is given the value 1. If the event does not happen, then y is given the value of 0. For example, if y represents whether a sports team wins a match, then y will be 1 if they win the match or y will be 0 if they do not. This is known as Binomial Logistic Regression. There is also another form of Logistic Regression which uses multiple values for the variable y. This form of Logistic Regression is known as Multinomial Logistic Regression. (source:wiki)

2.2 Tools Overview

2.2.1 Apache Flink

2.2.2 IDE

2.2.3 Standalone Cluster

2.3 Summary

Chapter 3

Methodology

3.1 Requirements

In order to run the project, you need to have a working Java 8 or 11 installation and download: Apache Flink 1.10.1 for Scala 2.12 + The flinkmancer.jar . Unpack the downloaded archive and run the following commands:
“ ./flink-1.10.0/bin/start-cluster.sh “
“ ./flink-1.10.0/bin/flink run flinkmancer.jar “
The datasets needs to be in ./data/layer for each different layer used.
This particular project uses 4 different layers, “follow”, ”reply”, ”quote”, ”retweet”.
After the program is executed , it produces a Features.csv file, containing 100 Features about each Edge of Vectors (each Vector being a user) and their common neighbors.

3.2 Design

This is a Maven Project written in Java using Apache Flink. It contains 3 .java files: Flinkmancer.java , Features.java , Node.java . Flinkmancer.java being the main class of the project, sets up the flink environement, reads the datasets and creates Node objects which contain the Id for each Node (each user) and an ArrayList of Sets of IDs for the incoming and outgoing neighbors of each node, on each layer. Then we cross the list of all nodes with themselves so we have a dataset of n*n size (n = number of total nodes).
Node.java is simply the class that implements the Node objects, and contains ways to access each object variables.
Features.java contains the feature extraction function. This function uses the dataset of all Nodes created in Flinkmancer.java to extract 100 features for each Node Edge. It returns an array of 102 columns, first 2 being the ID of nodes , and rest 100 the features .

Chapter 4

Results

4.1 Experiment 1

4.2 Experiment 2