Project Configuration Manual

Introduction

This guide is intended to provide the information you need to successfully install, configure, and run the project on your own system.

Project Summary:

Using a graph-based method, the Patent Validity Detection Project aims to predict the validity of patents. Patents are represented as nodes in a citation graph, and the project employs Graph Attention Networks (GAT) to analyse the connections between patents and predict their validity status.

Project Objectives:

- Create a machine learning model capable of predicting the validity of patents based on their citation graph.
- Utilise Graph Attention Networks (GAT) to capture patent relationships and attributes with precision.
- Provide a concrete illustration of the application of deep learning techniques to graph-based tasks.
- In the field of patent analysis, provide a practical application of artificial intelligence.

Importance of the Project:

Predicting patent validity is essential for both legal and innovative purposes. Patent offices, inventors, and businesses can make informed decisions regarding patent filing, licencing, and litigation with the aid of an accurate assessment of patent validity. By automating this process with AI, we increase efficiency and decrease the likelihood of expensive legal disputes.

The Configuration Manual serves to:

This manual serves as your guide for navigating the Patent Validity Detection Project's setup and execution. This manual will help you:

- Understand the necessary system requirements and dependencies for project execution.
- Install the necessary libraries and tools to properly configure the environment.
- Prepare data and incorporate it into the structure of the project.
- Execute the code and predict the patent's validity.
- Adjust model settings and parameters for optimal results.
- Examine typical problems that may arise during setup or execution.

By following the instructions in this manual, you will be able to easily run the project, explore its features, and tailor it to your specific requirements.

System Requirements

Before proceeding with the setup and operation of the Patent Validity Detection Project, ensure that your system meets the following hardware and software specifications:

Hardware Requirements:

- For faster calculations, a modern multi-core CPU (such as the Intel Core i5 or AMD Ryzen) is recommended.
- A minimum of 8 GB of RAM is recommended for efficient processing and instruction.

Software Prerequisites:

- The project is platform-independent and can be executed on Windows, macOS, or Linux.
- Since Python is used to develop the project, Python 3.x must be installed on your system.
- For the purpose of this project's code, Jupyter Notebook was used.

Mandatory Python Libraries:

The following Python libraries must be installed for the project to run: You can install them via code snippets or a virtual environment.

Pandas: A library for manipulating data

Numpy: A library for numerical computations

Scikit-learn: A library for machine learning

Scipy.stats: Statistics-related SciPy library functions

Matplotlib.pyplot: A library for plotting and visualising data

Seaborn: A library for statistical data visualisation

Networkx: A graph creation, manipulation, and analysis library

The sklearn.preprocessing method StandardScaler: A module for preprocessing feature scaling This module divides datasets into training and test sets.

Torch: The PyTorch library for deep learning

Torch.nn: The neural network module for PyTorch

Torch-scatter: A Python library for scatter operations

Torch-sparse: A PyTorch library for sparse tensor operations

Torch-cluster: A PyTorch Geometric Library for Clustering Operations on Graphs

Torch-spline-conv: A PyTorch Geometric library for convolutions based on splines

Torch-geometric: A PyTorch library for graph-based deep learning

Graph Attention Network layer from PyTorch Geometric

Sklearn.metrics: A module for evaluating classification metrics such as precision, recall, and F1-score

Ensure that these libraries are installed in your environment before successfully executing the code.

Installation notes:

- To isolate the project dependencies, it is recommended to use a virtual environment (such as virtualenv or conda).
- TensorFlow requires that the appropriate GPU drivers and CUDA toolkit be installed in order to support GPUs.

You will be able to set up and run the Patent Validity Detection Project after ensuring that your system meets these requirements and has the required libraries installed.

Commands to Install Dependencies

Here is a list of libraries and packages that your project depends on, along with their version numbers and installation instructions:

1. pandas (1.3.3)

! pip install pandas

2. numpy (1.21.4)

! pip install numpy

3. scikit-learn (0.24.2)

! pip install scikit-learn

4. scipy (1.7.3)

pip install scipy

5. matplotlib (3.4.3)

! pip install matplotlib

6. seaborn (0.11.2)

! pip install seaborn

7. networkx (2.6.3)

! pip install networkx

8. torch (1.8.0)

pip install torch

9. torch-scatter (latest)

! pip install torch-scatter -f https://pytorch-geometric.com/whl/torch-1.8.0+cu101.html

10. torch-sparse (latest)

! pip install torch-sparse -f https://pytorch-geometric.com/whl/torch-1.8.0+cu101.html

11. torch-cluster (latest)

! pip install torch-cluster -f https://pytorch-geometric.com/whl/torch-1.8.0+cu101.html

12. torch-spline-conv (latest)

! pip install torch-spline-conv -f https://pytorch-geometric.com/whl/torch-1.8.0+cu101.html

13. torch-geometric (latest)

pip install torch-geometric

Make sure to install these dependencies in your environment before running the project to ensure that all required packages are available.

Set up Google Colab Environment

Here are the detailed instructions for setting up the Google Colab (Jupyter Notebook) project environment. Obtaining access to Google Colab:

1. Launch your internet browser and go to Google Colab at https://colab.research.google.com/.
2. To create a new Colab notebook, select "File" > "New Notebook" from the menu bar.
3. Install the Necessary Packages:

Execute the following commands in a code cell to install the required packages:

!pip install pandas numpy scikit-learn scipy matplotlib seaborn networkx

!pip install torch
!pip install torch-scatter torch-sparse torch-cluster torch-spline-conv -f https://pytorch-geometric.com/whl/torch-1.8.0+cu101.html
!pip install torch-geometric

These commands will install the required Colab libraries and packages.

4. Import Dependencies:

Import the required libraries and packages in a code cell after installing the packages:
import pandas as pd
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns
import networkx as nx
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import torch
import torch.nn as nn
from torch_geometric.nn import GATConv
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score

5. Project code

You can now add your project's code to the cells that follow. This may involve loading data, defining the GAT model, defining the loss function, training the model, and assessing its performance.

6. Running Cells:

To execute each code cell, select it and press Shift+Enter. This will execute the code and display any output or errors in the output area of the cell.

7. Saving and sharing

After writing and executing your code, you can save the notebook to Google Drive or download it. You can also collaborate with others using the shared notebook.

Remember that Google Colab offers a convenient environment for GPU-accelerated Python code execution. If your project requires GPU support, ensure that you're connected to a Colab runtime that offers it.

You should be able to set up the project environment and begin working on your GAT model in Google Colab after completing these steps.

Data Preparation

Certainly, the following outlines how users may prepare their data for the project:

- Data Retrieval:

The project utilises the Google Public Patent Dataset, which is kept in Google BigQuery. Users must retrieve and export the dataset to their Google Drive.

- Exporting Data to Google Drive:

Follow the steps below to export BigQuery data to Google Drive:

    - Execute a SQL query against the Google Public Patent Dataset in BigQuery to retrieve the pertinent data.
    - Export the results of the query to a CSV or JSON file.
    - Save the exported file in a folder of your choosing on your Google Drive.
- Data Maintenance:

After exporting the data to Google Drive, it must be cleaned. The cleaning procedure may include:

    - Identification and management of missing data values. Either you can impute missing values or you can eliminate rows or columns with significant missing data.
    - Identify and eliminate data outliers using the appropriate techniques. You can choose to eliminate outliers if they are likely to affect the performance of your model.
    - Check for and eliminate any duplicate rows from your dataset.
- Exploratory Data Analysis (EDA):

Conduct exploratory data analysis to gain a deeper understanding of your data. This may involve creating data visualisations, calculating statistics, and recognising data patterns. EDA facilitates data comprehension and informs preprocessing decisions.

- Select the features that are pertinent to your project:

You can decide which features to include in your model using techniques such as correlation analysis, model feature importance, or domain knowledge.

- Loading data into project:

Once the data has been cleaned and prepared, it can be loaded into the project. If you're using a Jupyter Notebook (such as Google Colab), you can use Pandas to read CSV or JSON files exported to your Google Drive. Here is an example of a code fragment:

import pandas as pd

 # Define the path to the exported CSV or JSON file on your Google Drive

data_path = '/content/drive/MyDrive/your_data_folder/your_data.csv'

Load the data using pandas

data = pd.read_csv(data_path)

Replace 'your_data_folder' and 'your_data.csv' with the actual path to the data file you exported.

Users can retrieve the Google Public Patent Dataset from BigQuery, export it to their Google Drive, clean the data, perform exploratory data analysis, select relevant features, and load the data into their project environment for further analysis and model building using these steps.

Execution Of Code

Here are the steps required to execute the project code:

1. Environment Configuration:

Ensure that the environment has been configured according to the installation section of the configuration manual. If you are using a virtual environment, activate it.

2. Data Preparation:

Follow the "Data Preparation" section's instructions to retrieve the dataset from Google BigQuery, export it to your Google Drive, clean the data, conduct exploratory data analysis, and select relevant features.

3. Load Data and Libraries:

In your Jupyter Notebook, load the required libraries and the cleaned and prepared dataset. The code snippet provided in the "Data Preparation" section can be utilised.

4. Graph Construction:

Construct a citation graph network, and define your nodes and edges.

5. Define GAT Architecture

Define GAT Architecture Using Pytorch, define the architecture of the GAT model if you are working with it. Ensure that hyperparameters such as input_dim, hidden_dim, output_dim, and num_heads are specified.

6. Loss Function and Optimizer:

Specify the model's loss function and optimizer. You can use TensorFlow's built-in loss functions and optimizers with the GAT model.

7. Training Loop:

This requires iterating over a predetermined number of epochs, performing forward and backward passes, and updating model parameters with the optimizer. Use metrics such as loss and accuracy to track training progress.

8. Evaluation

After training, evaluate the performance of your model on a separate validation or test dataset. Calculate metrics including accuracy, precision, recall, and F1-score to evaluate the performance of the model.

9. Visualisation (Optional):

If applicable, present your results using visualisations. For this purpose, you can use libraries such as Matplotlib and Seaborn.

10. Execution:

Run each cell sequentially in your Jupyter Notebook. Ensure that you follow the correct execution order, as some cells may depend on the output of preceding cells.

11. Interpretation and Iteration:

Analyse the model's performance and interpret the results. If necessary, iterate on the model architecture, hyperparameters, and preprocessing steps to enhance the outcomes.

Remember to include comments and explanations for each step to improve the readability of your code. Debugging print statements and documentation can be helpful if you encounter errors or unexpected behaviour.

Following these steps and executing the code in a Jupyter Notebook environment will allow you to train and evaluate your model for detecting patent validity.

Troubleshooting

Here is a section on troubleshooting that addresses common user issues and provides steps to resolve them:

1. Model Training Losses Are Not Decreasing

Error Message: None

Potential causes: Ineffective model architecture, incorrect loss function, inappropriate hyperparameters, or insufficient training data are potential causes.

Steps for Troubleshooting:

- Verify that the model architecture is appropriate for the problem at hand.
- Check that the loss function is suitable for binary classification (such as binary cross-entropy).
- Experiment with various hyperparameters, including the learning rate, batch size, and number of epochs.
- Check the quantity and quality of your training data again.

2. Memory Errors or Sluggish Performance

Error Message: Exhaustion of system memory, programme freezes, or very slow execution

Potential causes: Large dataset size, inefficient data loading, or a lack of memory optimisation are potential causes.

Steps for Troubleshooting:

- Utilise generators or DataLoader classes for batch-wise data loading to optimise data loading.
- Reduce the size of the dataset or select a subset for testing purposes.
- If GPU acceleration is available, use it.
- In order to identify bottlenecks, profile your code.

3. Failure of Conda Environment Creation

Error Message: "EnvironmentCreationError" or similar conda errors.

Potential Causes: include package conflicts and incorrect dependencies.

Steps for Troubleshooting:

- Ensure that your package versions are compatible.
- Check your environment twice. yml file for errors in syntax.
- Use a new Conda environment to create the environment.

4. Problems with TensorFlow/Torch installation

Error Message: "ModuleNotFoundError"

Potential causes: Network issues, incompatible package versions, or missing dependencies are possible causes.

Steps for Troubleshooting:

- Verify that you are employing the appropriate installation commands for your environment (e.g., pip or conda).
- Check the version of the packages and ensure compatibility.
- If network problems persist, consider using alternative installation sources or mirrors.

5. Problem: Errors in Graph Creation or Loading

Error Message: Errors associated with loading or preprocessing graph data

Potential causes: Incorrect file paths, data format issues, or missing data files are potential causes.

Steps for Troubleshooting:

- Verify that the file paths are correct by double-checking them.
- Check the structure and format of your data files (CSV, JSON, etc.).
- Confirm that the specified locations contain the required data files.

6. Unsatisfactory Model Performance

Error Message: None

Possible Causes: Inadequate data preprocessing, insufficient feature engineering, and incorrect evaluation metrics

Steps for Troubleshooting:

- Ensure that data is appropriately preprocessed, including the handling of missing values and scaling features.
- Experiment with various data attributes or representations.
- Verify that you are employing the appropriate evaluation metrics for your problem (e.g., precision, recall, F1-score).

Remember that the process of debugging is iterative. If you encounter a problem, you should attempt to deconstruct it, identify potential causes, and test solutions incrementally. In addition, online forums, communities, and documentation can frequently provide insights and solutions to particular problems.