ALTEGRAD Challenge: Predicting the h-index of Authors

DaSciM team, École Polytechnique

January 2021

1 Introduction

The goal of this challenge is to study and apply machine learning/artificial intelligence techniques to a real-world regression problem. In this regression problem, each sample corresponds to a researcher (i. e., an author of research papers), and the goal is to build a model that can predict accurately the h-index of each author. More specifically, the h-index of an author measures his/her productivity and the citation impact of his/her publications. It is defined as the maximum value of h such that the given author has published h papers that have each been cited at least h times. To build the model, you are given two types of data: (1) a graph that models the collaboration intensity of two individuals (i. e., whether they have co-authored any papers), and (2) the abstracts of the top-cited papers of each author. The problem is very related to the well-studied problems of node regression and text regression. The pipeline that is typically followed to deal with the problem is similar to the one applied in any regression problem; the goal is to learn the parameters of a model from a collection of training samples (with known h-index values) and then to predict the h-index of individuals for whom this information is not available. In the context of this challenge, you are expected to combine large-scale data from different sources (i. e., graph data and textual data). This setting poses several challenges ranging from data cleaning/preprocessing to model design and hyperparameter tuning. Another issue that makes things even more complicated is that in most real-world learning problems, there is a lack of labeled data, and therefore the size of the training set could be much smaller than that of the test set.

The challenge is hosted on Kaggle[1], a platform for predictive modelling on which companies, organizations and researchers post their data, and statisticians and data miners from all over the world compete to produce the best models. The challenge is available at the following link: `https://www.kaggle.com/c/altegrad-2020`. To participate in the challenge, use the following link: `https://www.kaggle.com/t/9ac6db913ca5409c932bfe27f6d31d48`.

2 Dataset Description

As part of the challenge, you are given the following files:

1. **collaboration_network.edgelist**: a co-authorship network where nodes correspond to authors that have published papers in computer science venues (conference or journal) and two nodes are connected by an edge if they have co-authored at least one paper. The graph consists of $231,239$ vertices and $1,777,338$ edges in total. Each line in the file represents an edge, i. e., a pair

[1] `https://www.kaggle.com/`

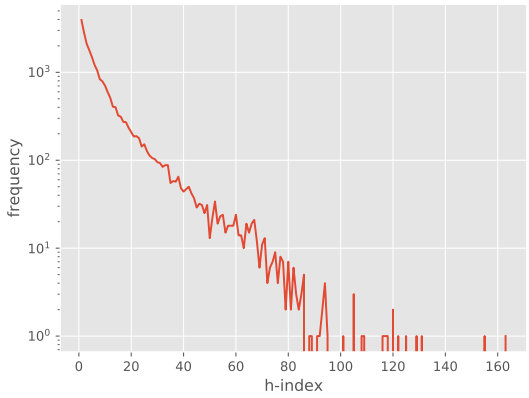

Figure 1: Distribution of h-index values of the authors of the training set.

of node IDs separated by a space character. The co-authorship network has been extracted from the Microsoft Academic Graph[2].

2. **author_papers.txt**: a list of authors and the IDs of their top-cited papers. Each row of this file is formatted as follows:

```
author_ID:[paper1_ID,paper2_ID,...,paperM_ID]
```

where "author_ID" is the ID of the author and "paper1_ID", "paper2_ID", "paperM_ID" are the IDs of his/her papers.

3. **abstracts.txt**: for each paper, this file contains the ID of the paper along with the inverted index of its abstract. Roughly speaking, an inverted index is a dictionary where the keys correspond to words and the values are lists of the positions of the corresponding words in the abstract. For example, the abstract "that is all that I know" would be represented as follows:

```
paper_ID----{"IndexLength":5,"InvertedIndex":
{"that":[0,3],"is":[1],"all":[2],"I":[4],"know":[5]}}
```

4. **train.csv**: it contains $23,124$ labeled authors. Each row of the file contains the ID of an author and his/her h-index.

5. **test.csv**: this file contains the IDs of $208,115$ authors. The final evaluation of your methods will be done on these authors and the goal will be to predict the h-index of each one of these authors.

Note that an h-index is a nonnegative integer. Furthermore, in general, the vast majority of the authors have relatively small h-index values, while only a few authors have very high h-index values. The distribution of the h-index values of the authors that belong to the training set is illustrated in Figure 1. Note that the vertical axis is in logarithmic scale.

[2] https://www.microsoft.com/en-us/research/project/microsoft-academic-graph/

3 Tasks and Evaluation

In the context of this challenge, you are asked to develop your own algorithms for predicting an author's h-index. You are expected to deal with all the different parts of the pipeline, e. g., cleaning and data preprocessing, generation of text and node representations, use of different features and regression algorithms, hyperparameter tuning, etc.

The performance of your models will be assessed using the *mean absolute error* (MAE). This metric is defined as the arithmetic average of the absolute errors. Specifically, it is defined as:

$$\text{MAE} = \frac{1}{N}\sum_{1}^{N}|\hat{y}_i - y_i|$$

where N is the number of test samples (i.e., authors), y_i is the h-index of the i^{th} author of the test set and $\hat{y}_i$ is the predicted h-index of that author. The objective is to achieve the smallest possible mean absolute error on the test set.

To evaluate your models, you need to create a `.csv` file consisting of the following two columns: (1) IDs of the authors of the test set, and (2) predicted h-index values. An example of how this file can be generated is shown in the provided scripts that implement the two baselines (see below).

4 Provided Source Code

You are given two scripts written in Python that will help you get started with the challenge. The first script (`text_baseline.py`) uses features extracted from the textual content of the abstracts of the authors' top-cited papers along with a Lasso regression model, and achieves a mean absolute error of 6.65 on the public leaderboard. The script makes use of two other scripts (`paper_representations.py` and `author_representations.py`) which map abstracts to vectors and combine the vector representations of the abstracts of each author's most cited papers to produce author representations. The second script (`graph_baseline.py`) uses solely graph-based features with a Lasso regression model for making predictions. This model achieves a mean absolute error of 7.24 on the public leaderboard. As part of this challenge, you are asked to write your own code and build your own models to predict the h-index of the authors. You are advised to use both textual information and features extracted from the graph.

5 Useful Python Libraries

In this section, we briefly discuss some packages that can be useful in the challenge:

- A very powerful machine learning library in Python is `scikit-learn`[3]. It can be used in the preprocessing step (e.g., for feature selection) and in the classification task (several regression algorithms have been implemented in `scikit-learn`).
- A very popular deep learning library in Python is `PyTorch`[4]. The library provides a simple and user-friendly interface to build and train deep learning models.
- Since you will deal with data represented as a graph, the use of a library for managing and analyzing graphs may be proven important. An example of such a library is the `NetworkX`[5] library

[3] `http://scikit-learn.org/`
[4] `https://pytorch.org/`
[5] `http://networkx.github.io/`

of Python that will allow you to create, manipulate and study the structure and several other features of a graph.

- Since you will also deal with textual data, the Natural Language Toolkit (`NLTK`)[6] of Python can also be found useful.
- `Gensim`[7] is a Python library for unsupervised topic modeling and natural language processing, using modern statistical machine learning. The library provides all the necessary tools for learning word and document embeddings.

6 Rules

Read carefully and make sure you follow the rules below. Teams breaking the rules will get penalized.

- **Team size** limit is 3 students. This limit is strict, no exceptions will be made.
- **Deadline**: all submissions must be uploaded by Tuesday, February 16, at midnight Paris time. No extension will be granted.
- **Submissions** must be uploaded here. Make sure that:
  - you upload **one submission per team**, as as a single archive named after the team
  - the archive contains **only**: (1) the team's code stored in a `/code/` subdirectory and (2) the team's report **as a .pdf file**
  - team name matches that on the Kaggle leaderboard
  - the **team name** and **the names of all team members** appear on the first page of the report
  - your submission does not contain the original data or the description of the competition
- **Reproducibility**: your code must allow the jury to reproduce your Kaggle submission.
- **Report format**: PDF reports **must follow the IJCAI format**: `https://www.ijcai.org/authors_kit`. In particular, they must have a maximum of six pages, plus at most one for references.
- **Questions**: all general questions **must be asked on the Kaggle forum of the competition**.
- **Reverse engineering**: your predictions must be generated only from the data provided.

7 Grading Scheme

Grading will be on 100 points total:

- **50 points** will be allocated to the presentation that you will give and your ability to answer the questions that you will be asked.
- **20 points** will be allocated based on the raw Kaggle performance, provided that the submission is reproducible. That is, using only your code and the data provided on the competition page, the jury should be able to train your final model and use it to generate the predictions you submitted for scoring.

[6] `http://www.nltk.org/`
[7] `https://radimrehurek.com/gensim/`

- The content of the report will be worth **20 points**. Regardless of the performance achieved, the jury will reward here the research efforts, creativity and experimental rigor put into your work.
- Finally, **10 points** will be allocated to the organization and readability of the code and report. Best submissions will (1) clearly deliver the solution, providing detailed explanations of each step, (2) provide clear, well organized and commented code, (3) refer to relevant research papers.