INF 554 Machine and Deep Learning
Data Challenge: H-index Prediction

November 2021

1 Introduction

The goal of this data 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. Your solution can be based on both supervised and unsupervised learning techniques. You should aim for a minimum Mean Squared Error (MSE) i.e., the MSE will be the loss function we use to evaluate your models. This data challenge is hosted on Kaggle as an in-class competition. In order to access the competition you must have a Kaggle account. If you do not have an account you can create one for free. The URL to register for the competition and have access to all necessary material is the following:

`https://www.kaggle.com/c/inf554-2021`

2 Dataset Description

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

1. **coauthorship.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 $217,801$ vertices and $1,718,164$ edges in total. Each line in the file represents an edge, i. e., a pair of node IDs separated by a space character. The co-authorship network has been extracted from the Microsoft Academic Graph[1].

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

2. **author_papers.txt**: a list of authors and the IDs of their top-cited papers (at most 5, but can be 0). 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**: contains $174,242$ labeled authors. Each row of the file contains the ID of an author and his/her h-index.

5. **test.csv**: contains the IDs of $43,561$ authors. The final evaluation of your methods will be done on these authors.

3 Tasks and Evaluation

As mentioned above, the task is to predict the test set 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.

Note that an h-index is a nonnegative integer. Furthermore, in general, the vast majority of the authors have relatively small h-index, while only a few authors have very high h-index. Assuming that the test set labels follow a similar distribution with the train set, you have to take this imbalance into account. Moreover, keep in mind that each author may have more than one paper and each paper contains multiple words. Hence learning word representations with a model like word2vec requires a strategy to combine all representations to a final author representation that can be used to indicate the author's text. Finally, the coauthorship graph can provide important insight on the impact of the author e.g. the majority of emminent scientists tend to have a large collaboration network, but this is not the case for every author in the test set.

The performance of your models will be assessed using the *Mean Squared Error* (MSE). This metric is defined as the arithmetic average of the squared absolute errors. It is commonly used for regression tasks. Specifically, it is defined as:

$$\text{MSE} = \frac{1}{N}\sum_{1}^{N}(\hat{y}_i - y_i)^2$$

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 error on the test set.

Note that the test set has been randomly partitioned into public and private. Scores on the leaderboard are based on the public set, but final scores (based on which grading will be performed) will be computed on the private set. This removes any incentive for overfitting the test set.

4 Provided Source Code

You are given two baseline scripts written in Python that will help you get started with the challenge. The first script (`dummy_baseline.py`) predicts all authors in the public test set to have the most common h-index from the train set and has an error of 221.16. The second script (`baseline.py`) uses solely graph-based features (degree and k-core centrality) with a Lasso regression model for making predictions. This model (graph metrics) achieves an error of 129.0. 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`[2]. 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`[3]. 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 prove to be important. An example of such a library is the `NetworkX`[4] library 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`)[5] of Python can also be found useful.
- `Gensim`[6] 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. An alternative to it is `FastText`[7].
- If you do not want to spend a lot of time to produce the word embeddings, you can use transfer learning. Download a pretrained set of word embeddings on GoogleNews[8], Glove[9] or Numberbatch[10]) and combine them using the abstracts.txt and the 'author_papers.txt' to create author representations.

6 Grading Scheme

Each team has to be composed of 2 or 3 students. No larger or smaller teams will be accepted. Please join and use the slack channel ("team-finding") to organise yourselves into groups of 3 or coordinate offline. Group choices need to be submitted to us by **Friday 26th November at 17:00** at the below link

[2] `http://scikit-learn.org/`
[3] `https://pytorch.org/`
[4] `http://networkx.github.io/`
[5] `http://www.nltk.org/`
[6] `https://radimrehurek.com/gensim/`
[7] `https://fasttext.cc/`
[8] `https://code.google.com/archive/p/word2vec/`
[9] `https://nlp.stanford.edu/projects/glove/`
[10] `https://github.com/commonsense/conceptnet-numberbatch`

https://forms.gle/b3k7VpyxAnHXkwwE8

Grading will be out of 100 points in total. Each team should deliver:

A submission on the Kaggle competition webpage. (20 points) will be allocated based on raw performance only, provided that the results are reproducible. That is, using only your code, the data provided on the competition page and any additional resources you are able to reference and demonstrate understanding of, the jury should be able to train your final model and use it to generate the predictions you submitted for scoring.

A zipped folder in moodle (30 points) including:

1. A folder named "code" containing all the scripts needed to reproduce your submission.
2. A README file with brief instructions on how to run your code and where it expects the original data files.
3. A report (.pdf file), of max 3 pages, excluding the cover page and references. In addition to your self-contained 3-page report, you can use up to 3 extra pages of appendix (for extra explanations, algorithms, figures, tables, etc.). Please ensure that both your real name(s) and the name of your Kaggle team appear on the cover page.

The 3-page report should include the following sections (in that order):

- **Section 1: feature selection/extraction (15 points).** Independent of the prediction performance achieved, the jury will reward the research effort done here. Best submissions will capture both graph and text information. You are expected to:
  1. Explain the motivation and intuition behind each feature. How did you come up with the feature (e.g., are you following the recommendation of a research paper)? What is it intended to capture?
  2. Rigorously report your experiments about the impact of various combinations of features on predictive performance, and, depending on the regressor, how you tackled the task of feature selection.
- **Section 2: model tuning and comparison (10 points).** Best submissions will:
  1. Compare your model against different regression models (e.g., Neural Network, Support Vector Regression, Random Forest...).
  2. For each regressor, explain the procedure that was followed to tackle parameter tuning and prevent overfitting.

Report and code completeness, organization and readability will be worth **5 points**. 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 research papers. You are free to search for relevant papers and articles and try to incorporate their ideas and approaches into your code and report as long as (a) it is clearly cited within the report, (b) it is not a direct copy of code and (c) you are able to demonstrate understanding of the content you incorporated.

An oral presentation of your project and the achieved results (50 points) Oral presentations will be scheduled in the week of the 13th December. A schedule on which you can choose presentation slots will be released after the team submissions have been made. We ask you to prepare 15 minute presentations, which will then be followed by questions from the examiners. It is required that all group members are present and take a speaking role during the presentation.

7 Submission Process

Submission files should be in .csv format , and contain two columns respectively named ”author” and ”hindex”. The ”author” column should contain the author id as given in the dummy_baseline.py. The hindex” column should contain the predictions (no negative integer). Note that a sample submission file is available for download (dummy_submission.csv) . You can use it to test that everything works fine. The competition ends on **Wednesday 8th December at 17:00**. This is the deadline for you to submit a compressed file containing your source code and final report, explaining your solutions and discussing the scores you have achieved. Until then, you can submit your solution to Kaggle and get a score at most 5 times per day. There must be one final submission per team.