INF 554 Machine and Deep Learning

Data Challenge: H-index Prediction. Team NLD

Leder Aguirre[1], Denis Mbey Akola[2], and Nicolas Espina[3]

[1]M1 Computer Science-Optimisation, leder-yhivert.aguirre-ramirez@polytechnique.edu.
[2]M1 Cyber-Physical Systems, denis.akola@ip-paris.fr
[3]M1 Applied Mathematics,nicolas.espina-quilodran@polytechnique.edu

December 9, 2021

Abstract

The goal of this data challenge is to predict accuractely the h-index of scientific authors in computer science using Machine Learning techniques. The dataset consist of: (i) a collaboration graph where each node represents an author and edges between two nodes indicates a coauthorship on at least one paper, (ii) a list of the top-cited papers of some of these authors, and (iii) a list of abstracts of some of these papers. 75% of the data set is dedicated to the training phase and the metric to evaluate each model is the Mean Squared Error (MSE). For all the approachs proposed in this project, we construct an author embedding from the abstracts, another from the collaboration network and also structural features of that graph. Our best model achieved a score (MSE) of 49.64101 and ranked 11 on Kaggle private leaderboard. All deliverables for this project are available on Github.

1 Introduction

The problem of predicting the h-index of an author belongs to the class of supervised learning problems. To solve this kind of problems, there is a plethora of models proposed in the literature. We can classify them into two big groups: naive approaches like lasso regression and k-Nearest Neighbours, tree-based like random forest and neural network-based like deep neural networks. Thus,considering this problem, we decided to use both regression approaches and deep neural networks to predict the h-index of the authors. We therefore trained several regression models and a deep neural network. For the regression task, we trained the model using Random Forest, Lasso Regressor, LightGBM, Xgboost, and K-Nearest Neighbors.The deep neural network was based on a recurrent neural network architecture.

2 Data preprocessing and Analysis

The data that was provided for this challenge included the inverted abstracts of papers the authors wrote, the co-authorhsip network graph, and the list of authors and their top cited papers. To a get a good model that works better on this prediction task ,we spent a great amount of time cleaning and preparing this data. First, we converted the abstract from the inverted format to normal text strings without the indexes. Upon further analysis of the abstract, we discovered that not all the abstracts were in english. About 99.2% of the abstracts provided were in english,however,the remaining abstracts were in other languages (French,Spanish,German,Portuguese etc). In oder to classify the abstracts into other languages, we decided to use a threshold which was computed based on the number of non-english words which were found in a single abstract. A threshold of 0.50 was used in this exercise. It is common to have some words from foreign languages infiltrated into other languages so setting a threshold help us to avoid classifying every abstract that had a foreign word which might have been adopted into English to be a foreign abstract.

We leveraged the Google translator python api to convert all these abstracts to English. We used NLTK library to remove stopwords.

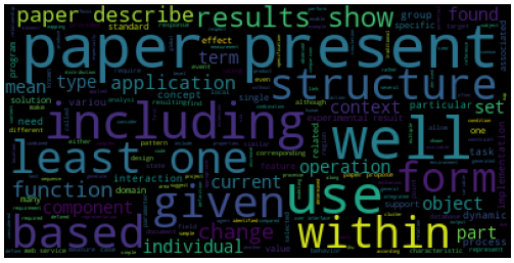

Figure 1: Wordcloud chart created from the abstracts

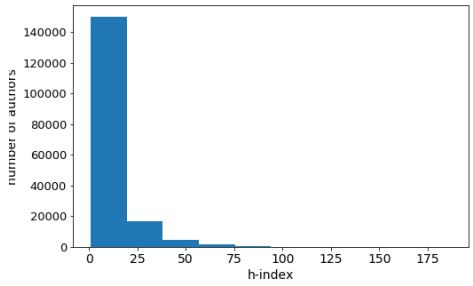

Figure 2: Histogram of the h-index.

Also,we took advantage of WordCloud Python package to visualise the abstracts data and we discovered some more words which were frequent in all abstracts and we likewise treated them as stopwords. A sample of the visualisations with wordcloud is shown in Figure 1:wordcloud above.

The collaboration graph was modified to obtain a weighted version (weights based on the number of top-citer papers co-authored). This was created with the Networkx package, so that the third column is a dictionary where the parameter name is the key. A lighter version was also manually calculated to have only the weight value in the third column.

3 Data analysis

4 Feature selection and feature extraction

From each input data a group of features is extracted and used in the general model. A brief description of such feautures is presented below and detailed later.

- We calculate some structural feature from the collaboration graph along with the list of top-cited papers.
- Additionally, we determine the author embeddings from the nodes that it represents in the collaboration graph.
- In the same way, we determina the paper embeddings from the abstracts. Taking the average on these vectors we get another vector representation of the authors.

4.1 Structural graph features

Since there are other alternative indices but correlated with the h index, for the first version of this file we consider the following structural features of a graph:

1. Core number: The core number is an indicator of the largest value k of a k-core containing that node.This was computed using the networkx core submodule based on algorithm adopted from this paper [1].
2. Clustering coefficient: The clustering coefficient is a measure of the degree to which nodes in a graph tend to cluster together.The clustering clustering coefficient for a vertex is given by a proportion of the number of links between the vertices within its neighbourhood divided by the number of links that could possibly exist between them.This provides a measure of how tightly knit the graph nodes are.
3. Centrality: The centrality of a graph node provides useful information about the relationship that exist between nodes.It is computed as the average length of the shortest path between a node and all other nodes in a graph.
4. Page rank:Page rank provides a measure of importance to each node in the graph.PageRank computes a ranking of the nodes in the graph G based on the structure of the incoming links.This is implemented in Networkx based on the algorithm presented in [5] by it's authors.

The last three features were also calculated for a the weighted graph where the weights represented by the co-authorship of a top-cited article (this allowed zero value weights, which is why the Simulated Walks of the Pecanpy.Node2vec package was manually modified to avoid node deletions during the node embedding computation). For the version two we add the following features:

5. Degree: The degree of a graph node depicts the number of nodes adjacent to it.
6. Onion number: This was computed based the onion decomposition of a network graph. This is implemented in Networkx.

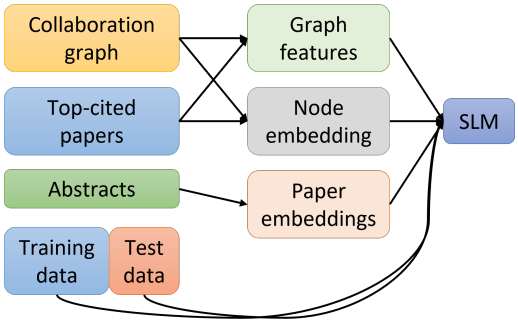

Figure 3: Methology followed for all the Supervised Learning Models (SLM). Each arrow represents the computing and information flux.

4.2 Node embedding

In this step, we used Node2Vec from Pecanpy to calculate the node embeddings. This implementation cache-optimized compact graph data structures and precomputing/parallelization to result in fast, high-quality node embedding.[4]

We specified a vector dimension of 20 and tried to apply dimension reduction using PCA, but the variance was almost uniformly distributed. Furthermore, when trying to group the vectors in two dimensions using t-SNE, no pattern was observed with respect to the distribution of the h-index. Experimenting with a larger vector dimension was left for future work.

4.3 Abstract Embeddings

In this step,we used Doc2Vec from gensim NLP Python Library to transform our processed abstracts into paragraphs or docs embeddings. Doc2Vec is an NLP tool for representing documents (text corpus) as a vector. It is a generalisation of word2Vec [3]. After that, the author embedding is obtained by averaging the paragraph vectors.

5 Models

We successfully use supervised learning techniques on a set of processed data. We tested six different models and did the parameter tuning on two of them.

5.1 Methodology

All the steps in the last section is summarize in the diagram of Figure 3.

5.2 Model tuning and comparison

We first try the run all the models with parameters close to the default value. The results are shown in Table 1.

Model regressor	MSE train	MSE test
KNN with k=7	111.269	119.863
LassoCV	77.026	75.995
FFNN	55.897	63.690
Random Forest	8.869	62.594
XGBoost	46.192	59.648
LightGBM	14.230	51.689

Table 1: Comparison of the models before parameter tuning.

The performance of the first two models gives us a new baseline for the other models in this section. The Feed-Forward Neural Network (FFNN), constructed as a set of sequential layers, don't give us satisfactory results but the execution is fast using the GPU. Finally, since Random Forest had the lowest MSE test compared to the formers, we tested more powerful models derived from Random Forest. So, the improvements with XGBoost and with LightGBM only confirms the theoretical progress on this field of Gradient Boosting model.[2]

In the two tables below we ilustrate the results for the parameter tuning of the FFNN and the LightGBM.

LR	DR	Act. function	UHL	MSE
0.01	0.1	LeakyReLU(0.1)	20,4	60.08
0.10	0.1	LeakyReLU(0.1)	20,4	60.25
0.10	0.1	ReLU()	20,4	61.16
0.01	0.1	ReLU()	25,5	62.95
0.01	0.2	Tanh()	25,5	64.94
0.10	0.2	Tanh()	20,4	67.16
1.00	0.2	LeakyReLU(0.1)	20,4	67.98
1.00	0.1	Tanh()	25,5	71.75

Table 2: Best results of parameter tuning for the Feed-Forward Neural Network. Nomenclature used: LR: Learning Rate, DR: Dropout Rate, UHL: Units in Hidden Layers. The complete table contains 72 scenarios with 3 LR, 2 DR, 2 UHL, 3 Act. functions and 2 validation size (0.25 turned out to be better).

From the Table 2 we concluded that LeakyReLU is the best activation functions compared to ReLU and Tanh. Furthermore, the lower the learning rate, the lower the MSE, keeping the other parameters constant.

From the Table 3 we concluded that a big number of splits and estimators contributed to a lower MSE. In contrast to the Neural Network model, a higher learning rate is preferred here, 0.05 to be exact.

Learning rate	Splits	Estimators	MSE
0.05	7	10000	49.434
0.05	7	8000	49.630
0.05	7	6000	49.976
0.05	7	6000	50.214
0.05	5	6000	50.591
0.025	7	6000	50.787

Table 3: Best results of parameter tuning for the LightGBM model. Here *splits* is the number of folds for the StratifiedKFold cross-validator and *estimators* is the number of gradient boosted trees.

6 Discussion

After several experiments,the LightGBM models performs better than the recurrent neural network and the other regressors. Thus,the LightGBM model is best for the prediction task for the h-index prediction.

Conclusions

One of the major setbacks which could account for why our model had a best MSE of 49.641 instead of a lower value could be attributed to the text processing and features extraction. For the text processing, we could have gone further, correcting manually some of the abstracts with errors and used a better translator. Regarding feature extraction, though we get a paper embedding with the abstracts, we could have added other features like a catagorical variable that assigns an author to a set of topics or try to mimic alternative indicators to h-index like Individual h-index or Field-weighted Citation Impact (FWCI).

References

[1] Vladimir Batagelj and Matjaz Zaversnik. "An O(m) Algorithm for Cores Decomposition of Networks". In: *CoRR* cs.DS/0310049 (2003). URL: http://arxiv.org/abs/cs/0310049.

[2] Guolin Ke et al. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[3] Quoc V. Le and Tomás Mikolov. "Distributed Representations of Sentences and Documents". In: *CoRR* abs/1405.4053 (2014). arXiv: 1405.4053. URL: http://arxiv.org/abs/1405.4053.

[4] Renming Liu and Arjun Krishnan. "PecanPy: a fast, efficient and parallelized Python implementation of node2vec". In: *Bioinformatics* 37.19 (Mar. 2021), pp. 3377–3379. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btab202. eprint: https://academic.oup.com/bioinformatics/article-pdf/37/19/3377/40556655/btab202.pdf. URL: https://doi.org/10.1093/bioinformatics/btab202.

[5] Lawrence Page et al. "The PageRank Citation Ranking : Bringing Order to the Web". In: *WWW 1999*. 1999.