PROJECT REPORT INF554

Data Challenge : H-index prediction

Du 08 Decembre 2021

Par:
FOTSO KAPTUE Paul Laudrup:
paul.fotso-kaptue@polytechnique.edu
DJOMENI DJIELA Wilfried:
wilfried.djomeni-djiela@polytechnique.edu
NGATCHA BAKOUE Didier Stéphane:
didier.ngatcha-bakoue@polytechnique.edu


l'
ÉCOLE
POLYTECHNIQUE
IP PARIS

SUMMARY

1 INTRODUCTION

In this report we will present you our work on the data challenge : h-index prediction.The h-index of an researcher 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. the data provide for this challenge are : a graph of collaboration between researchers and the abstracts of the top cited papers of each researchers. This report show how we extract the features from the data provided and how we build and fine-tuned ours models in other to build a predictor as accurate as possible. Accuracy in this case is measures by the Mean-Square-Error(MSE).

2 FEATURES EXTRACTION

2.1 Graph features

We first focused on coming up with relevant graph features that would characterize the scientific success of an author. According to [] there are two approaches to social network analysis : either study egocentric co-author networks, or the whole networks. The idea behind egocentric network is to look out, for a given node on the graph the characteristics and interactions between its neighbors and assessing from these the values that we want to predict. Whole network analysis studies those interaction between nodes in a geographically bounded zone.

We retained a certain set of variables and used these variables as our graph features and a matrix with these attributes.

2.1.1 • Features

Network Size : The h-index of an author is most likely to be high if the number of co-authors to whom he is connected is high. This observation was made in [] and confirmed while examining some sample authors with large h-indices (typically higher than 70) [1].

Number of Components/communities : This feature comes from the idea that high h-index authors often work with multiple different scientific communities. A community here consists of at least 3 authors linked together in the graph.So these communities never publish together and are linked solely by the edge-author. Number of isolates : As the number of isolates increase the h-index will increase.

Page Rank : This feature computes the ranking of a vertex in a graph, meaning its importance. [2]

$$PR(u) = \sum_{u \in \mathbf{N}(v)} \frac{PR(u)}{deg(u)}$$

Core number : A k-core is a sub graph of a graph in which all nodes have at least k neighbors.[2]

Average and maximum h-index of authors : Intuitively, we hypothesized that high h-index authors may cite each other, creating in the graph an accumulation of high h-index authors. This was called in [1] the Godfather Effect.

H-index of most eigen-vector central co-author : Eigenvector centrality is a measure of the social position of a node that uses the valued data. It is focused more on position within the entire network structure rather than the local network structure.Larger values of this variable yields higher h-indexes.[1]

2.1.2 • Additional features and results

We represented the correlation matrix between the previous variables and the h-indexes.

During, our implementations we tried adding some additional features inherited from the Node2Vec library which implement a Node2vec embedding.

The results as we will show in the a following section, were quite similar and slightly worse than the results obtained without these additional features. This means that the essential information of the graph structure was already captured by the above features.

2.2 Abstracts Features

For the features related to the abstracts we used several different methods, compared our results of our predictions on a validation set using only these features. The main idea was to come up with a representation of the authors using all the top cited abstracts of an author.

2.2.1 • Using word2vec pretrained data

In this section, we used the pretrained data from word2vec. We formatted the abstracts for each authors in order to remove punctuation and stop-words, and embedded each word of those abstracts using the embedding provided by word2vec. To obtain the representation of an author we simply computed the mean of these word embedding over all the words present in the abstracts.

But a major downside to this method, was the fact that many of the words present in the abstracts, most of them technical words deeply related to the computer science field, were absent from the dictionary of words in word2vec. Hence, our representations were incomplete, and did not fully capture the content of the abstracts. Thus, we obtained unsatisfactory results.

2.2.2 • Using Sentence-BERT

In this section we used a pretrained model name sentence-BERT from the sentence transformers framework. Sentence transformers is a python framework for state-of-the-art sentences, text and image embeddings. The initial work is describe in the paper[3]. We can use this framework to compute sentence/text embeddings for more than 100 languages. These embeddings can then be compared e.g. with cosine-similarity to find sentences with a similar meaning. This can be useful for semantic textual similar, semantic search, or paraphrase mining. In our case we were interested in paraphrase mining ; this is why we had used a pretrained model specific for this task.Unlike other embeddings methods, that are doing word embeddings, this methods makes paragraphs embeddings, which allows us to keep the context of each paragraph. contrary to what we thought ;the use of features from sentence-BERT combined with those from the graph gave us poorer results than those obtained by using features from the training of a word2vec network. This can be explained by the fact that training a word2vec network on our own data allows us to better contextualise the representation of our abstracts in our use case than using a pre-trained model that has learned on a wide variety of data.

2.2.3 • Training our word embedding models using word2vec

Word2vec weight with tf-idf : We trained Word2vec with the abstract dataset with purpose of been more contextual. In fact we realize that some words that are relevant for topic representation of an author are not present in word2vec defaults models (for example ipv, a word directly linked to telecommunication's protocols were not present in 'word2vec-google-news-300') . The trained word2vec have been weighed with tf-idf of the abstracts to compute abstract vector. The advantage of tf-idf is that it gives to every word a weigh relative to his importance in all the sentences. This approach has proved to be better than just a mean of word2vec sentence because it leverage both the word2vec embedding and the tf-idf method.

3
MODEL TUNING AND RESULTS

In an early phase of our work we tried using a polynomial regression using only the graph structure. Increasing the degree of the regression did not really affect the quality of the predictions. These results poor confirmed the fact that there are no linearity or simple relations between the features and the h-indexes.

Next, we implemented successively the 3 methods above, and adopted the best one for optimization. Additionally, we used Graph Neural Networks for our problem with 2 hidden layers of dimensionality 64 each.

For our MLP, the hyper-parameters we tuned were the dropout rate, the activation function, and the number of epochs of the training process of the model. The dropout rate was set to 0.1 and the activation function used was ReLU. The number of hidden layers tried were 2 hidden

layers of dimensionality 64, then one hidden layer of dimensionality 100. The latter performed slightly better than the first architecture.

For our XGBoost model,the relevant hyper-parameters we tuned were de n_estimator which represent the number of decision-tree in our model. we have test n_estimator in range(0,1000,50) in order to find the optimal number of tree which achieve the best performance while preventing over-fitting,because Xgboost over-fit a lot.

3.1 Results

method	MSE with Text features	MSE with Graph features	MSE with all features
XGBoostRgressor	75,45	77,12	53,40
MLPRegressor	71,88	81,53	52,72
GCNRegressor	78,24	67,34	56,43

Figure 1 – Performance of the different methods in the -index prediction task

RÉFÉRENCES

[1] Allison Hopkins Christopher McCarty, James W. Jawitz and Alex Goldman. Predicting author h-index using characteristics of the co-author network. In *EMNLP 2019*, 2012.

[2] Iakovos Evdaimon Giannis Nikolentzos, George Panagopoulos and Michalis Vazirgiannis. Can author collaboration reveal impact ? the case of h-index. In *EMNLP 2019*, 2021.

[3] Nils Reimers and Iryna Gurevych. Sentence-bert : Sentence embeddings using siamese bert-networks. In *EMNLP*, 2019.