INF554 – DATA CHALLENGE

JÉRÉMIE DENTAN, LOUIS PROFFIT, PAUL THÉRON

INTRODUCTION

The main difficulty of this project is related to the diversity of the datasets. Indeed, the aim was to predict the h-index of researchers based on three different datasets: *coauthorship.edgelist*, which provides a common publication graph on the authors, *abstracts.txt* which provides the abstracts of the authors' papers, and *author_paper.txt* which links the authors to their abstracts from the second dataset.

To address this challenge, we have chosen the approach detailed in figure 1. The main idea of this approach, inspired by [2], is to build a featured graph whose nodes are the authors, whose link are the co-authorships, and whose features combines features from the graph and from the abstracts. Then, we use GCN to predict h-indexes from this featured graph. The four key steps of this approach are the following:

- **Feature extraction from the abstracts:** we use different methods to extract embeddings from the abstracts: *word2vec*, *doc2vec* and *RoBERTa* (which is a higher-level method, with pre-trained parameters). These three methods will lead to three different h-index predictions.
- **Feature extraction from the graph:** we add to each node both high-level features and a *node2vec* embedding. The high-level features are for example the core number, the degree, and the number of papers. The *node2vec* embedding contains relevant information about the spatial structure of the graph.
- **Prediction with a neural network:** we train some 2-layers GCN to predict the h-indexes based on the three featured graph we have built.
- **Postprocessing:** we use a linear regression to combine our four h-index predictions. After that, we have some postprocessing: for example, we limit to 5 the h-index of researchers who have less than 5 papers in the dataset.

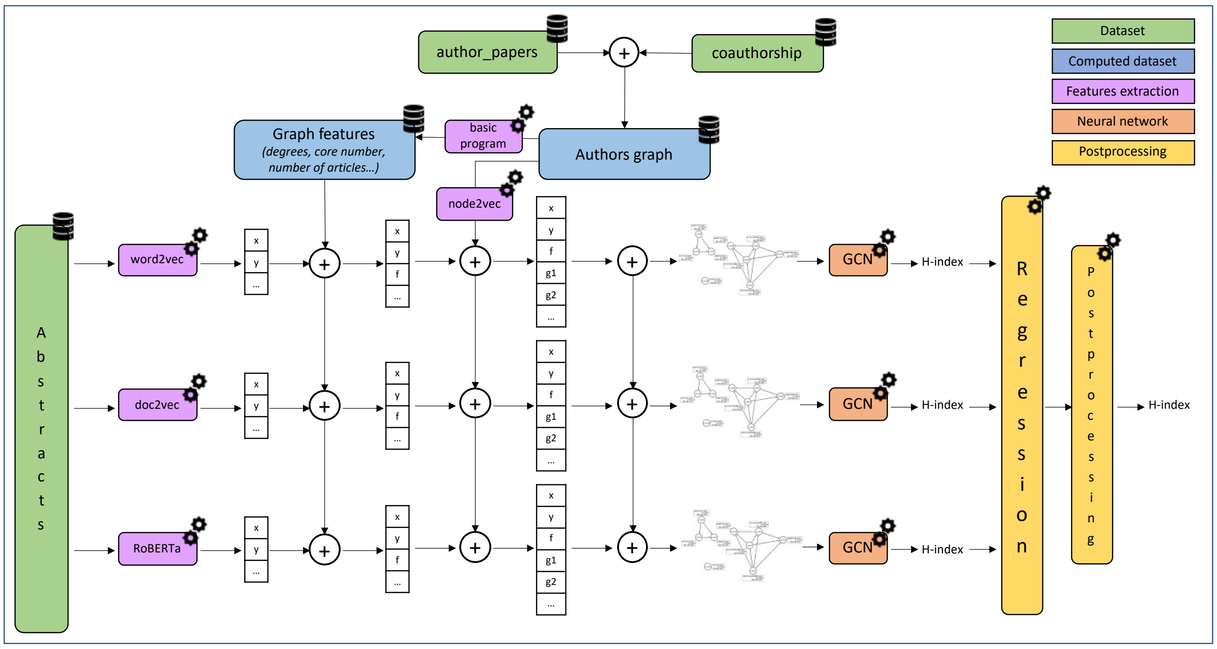

Figure 1: global view of our approach

I. Features extraction

We used feature extraction for three different purposes: to embed the abstracts, to add high-level features from the graph, and to add spatial features from the graph.

1. Embeddings of the abstracts

We need an embedding of the abstracts for the featured graphs that will be used to train the GCN.

Preprocessing

We first perform a serious preprocessing step, which aims to focus only on the core of the abstracts. We build the whole abstracts from the dataset, put in in lower case, remove punctuation and stop words, remove special characters… Some abstracts are empty (for example if they are written in Chinese or if they are not in the dataset), so we replace them with the word "science", which is neutral in that context.

Three methods

We use three methods to embed the abstracts:

- First, *word2vec* really focuses on the main ideas and on the topics of the abstracts since it focuses on the words themselves and their interactions. This is important since h-indexes are higher in some research fields than others, and some specific words such are 'outperform' can suggest high h-indexes.
- Second, *doc2vec* focuses on the document as a whole and its deep meaning, and not only on the words. This embedding was efficient for the validation set, but **paradoxically it obtained worse results for the test set.**
- Eventually, we use RoBERTa since it has proven very strong results in NLP [1].

In each case, we associate to a researcher **a combination of the embedding of all his or her articles,** giving more importance to the first article, then the second, etc. (coefficients: 2.0, 1.8, 1.6, 1.4, 1.2).

These methods have their advantages and disadvantages, so we build three different featured graphs, and combine the predictions at the end with a linear neural network.

2. High-level features from the graph

Some features are important for h-index prediction but are not taken into account by classic GCN such as *Pytorch Geometric's* ones. Those features are simple to add but have enabled us to gain about 15 points for the MSE loss:

- **Degree:** this is obviously linked to h-indexes.
- **Core number:** the h-indexes are the core number of the whole citation graph, so the core number of the partial graph is probably linked to the real h-index.
- **Centrality degree:** the more central a researcher is, the greater his or her h-index.
- **PageRank** (by *Google*): this provides a score of credibility in the graph, probably correlated to h-index since a more credible papers are more likely to be cited.
- **Onion layer** (from *networkx*): it refines the core number by providing information on the internal organization of each k-shell.
- **Number of articles:** the more the researcher writes article, the greater the h-index.

3. Node2vec

Node2vec enables to capture several features from the spatial configuration of our vertices in the graph. Indeed, GCN (that we will use lately) can efficiently leverage the features of the neighbors of a node. However, the random walks of *node2vec* can efficiently provide features about how a node interacts with the other on a larger scale.

II. Model tuning

1. A first attempt

In a first attempt, we tried to add the h-index as a node feature for the node with a known h-index (and the mean for unknown h-indexes). **The goal was that the GCN learn to use the h-index of neighbors to predict the h-index of unknown nodes.** We trained the network on the whole train set, but the backpropagation was for the validation set only. Indeed, if the backpropagation had been for the train set as well, the GCN would only have learned to predict its own h-index, which is useless. Unfortunately, this approach only provided us a 90 MSE loss, so we abandoned it.

2. GCN

We have used GCN for our h-index prediction since they have proven strong results for h-index prediction in [2]. For the number of layer and the activation function, we used the results from [3], with a two-layer GCN and a ReLU activation. Indeed, [3] shows that there is no need to add more than 2 layers, and in practice our neural networks worked bad with more than 2.

The size of the inner layers is 64 and 32, since we empirically have observed good results like this.

3. Linear model

At the end, we combine the three h-indexes predicted with a *scikit-learn* fully linear 1-layer network to have a better prediction, taking into account the different embeddings.

III. Practical results

In practice, some of our implementations unfortunately did not work, which we detail here:

- ***RoBERTa:*** We did not achieve the *RoBERTa* embedding, due to a memory shortage, so we only relied on *word2vec* and *doc2vec.*
- ***Node2vec:*** We did not achieve to run *node2vec*, for an unknown reason (probably linked to an incorrect use of the library but we could not fix that). Hopefully, the GCN alone was enough for an acceptable precision.

In practice, we have obtained our best results (74.20) with *word2vec* only, with 64 and 32 neurons in the hidden layers, because *doc2vec* was paradoxically less efficient on the test set than on the validation one, so the combination of the two approaches did not improve our score.

IV. References

[1] *RoBERTa: A Robustly Optimized BERT Pretraining Approach*, https://arxiv.org/abs/1907.11692v1

[2] *Can Author Collaboration Reveal Impact? The Case of h-index*, https://arxiv.org/abs/2104.05562

[3] *Semi-supervised Classification with Graph Convolutional Networks*, https://arxiv.org/abs/1609.02907